PROGRAMMABLE CIRCUIT AND RELATED COMPUTING MACHINE AND METHOD

Publication number: KR20050086424 (A)
Publication date: 2005-08-30
Inventor(s): RAPP JOHN W [US]; JACKSON LARRY [US]; JONES MARK [US]; CHERASARO TROY [US]
Applicant(s): LOCKHEED CORP [US]
Classification:
- international: ***G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/78;*** **G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/76;** (IPC1-7): G06F9/445
- European: G06F9/38S4
Application number: KR20057007751 20050430
Priority number(s): US20030683929 20031009; US20030683932 20031009; US20030684053 20031009; US20030684057 20031009; US20030684102 20031009; US20020422503P 20021031

Also published as:
WO2004042560 (A2)
WO2004042560 (A3)
WO2004042574 (A2)
WO2004042574 (A3)
WO2004042569 (A2)
more >>

Abstract not available for KR 20050086424 (A)
Abstract of corresponding document: **WO 2004042560 (A2)**

A peer-vector machine includes a host processor and a hardwired pipeline accelerator. The host processor executes a program, and, in response to the program, generates host data, and the pipeline accelerator generates pipeline data from the host data. Alternatively, the pipeline accelerator generates the pipeline data, and the host processor generates the host data from the pipeline data. Because the peer-vector machine includes both a processor and a pipeline accelerator, it can often process data more efficiently than a machine that includes only processors or only accelerators. For example, one can design the peer-vector machine so that the host processor performs decision-making and non-mathematically intensive operations and the accelerator performs non-decision-making and mathematically intensive operations.; By shifting the mathematically intensive operations to the accelerator, the peer-vector machine often can, for a given clock frequency, process data at a speed that surpasses the speed at which a processor-only machine can process the data.

(19) KOREAN INTELLECTUAL PROPERTY OFFICE

KOREAN PATENT ABSTRACTS

(11) Publication number: 1020050086424 A
(43) Date of publication of application: 30.08.2005

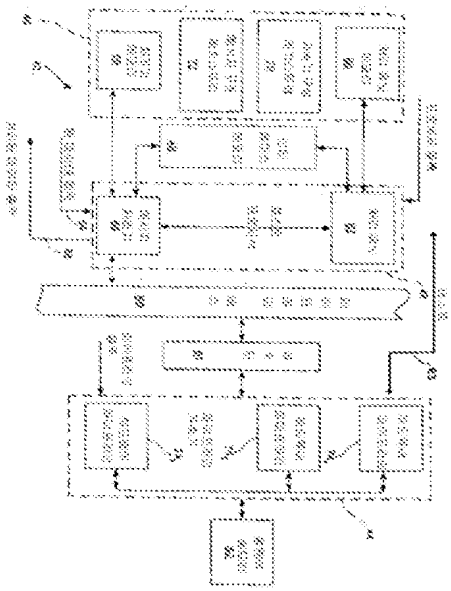
(21) Application number: 1020057007751
(22) Date of filing: 30.04.2005
(30) Priority: 2003 683929 US 09.10.2003
(51) Int. Cl: *G06F 9/445 (2006.01);*

(71) Applicant: LOCKHEED MARTIN CORPORATION
(72) Inventor: RAPP JOHN W.
JACKSON LARRY
JONES MARK
CHERASARO TROY

(54) **PROGRAMMABLE CIRCUIT AND RELATED COMPUTING MACHINE AND METHOD**

(57) Abstract:

A programmable circuit receives configuration data from an external source, stores the firmware in a memory, and then downloads the firmware from the memory. Such a programmable circuit allows a system, such as a computing machine, to modify the programmable circuit's configuration, thus eliminating the need for manually reprogramming the configuration memory. For example, if the programmable circuit is an FPGA that is part of a pipeline accelerator, a processor coupled to the accelerator can modify the configuration of the FPGA. More specifically, the processor retrieves from a configuration registry firmware that represents the modified configuration, and sends the firmware to the FPGA, which then stores the firmware in a memory such as an electrically erasable and programmable read-only memory (EEPROM). Next, the FPGA downloads the firmware from the memory into its configuration registers, and thus reconfigures itself to have the modified configuration.



This Facsimile First Page has been artificially created from the Korean Patent Abstracts CD Rom

(19)대한민국특허청(KR)
(12) 공개특허공보(A)

(51) 。 Int. Cl.[7]
G06F 9/445

(11) 공개번호 10-2005-0086424
(43) 공개일자 2005년08월30일

(21) 출원번호 10-2005-7007751
(22) 출원일자 2005년04월30일
번역문 제출일자 2005년04월30일
(86) 국제출원번호 PCT/US2003/034656
국제출원일자 2003년10월31일
(87) 국제공개번호 WO 2004/042569
국제공개일자 2004년05월21일

(30) 우선권주장
10/683,929 2003년10월09일 미국(US)
10/683,932 2003년10월09일 미국(US)
10/684,053 2003년10월09일 미국(US)
10/684,057 2003년10월09일 미국(US)
10/684,102 2003년10월09일 미국(US)
60/422,503 2002년10월31일 미국(US)

(71) 출원인
록히드 마틴 코포레이션
미국 버지니아주 20110, 매나사스, 굴원 드라이브 9500, 메일 드롭 043, 빌딩 400

(72) 발명자
웹, 존 더블유
미국 버지니아 20110, 매나사스, 리버 크레스트 로드 9350
잭슨, 데릭
미국 버지니아 20112, 매나사스 크레스트북 드라이브 13093
존, 마크
미국 버지니아 20120, 센트레빌, 오크에어 플레이스 15342
케라사프, 프로이
미국 버지니아 22701, 컬페퍼, 캐스트왈 코트 1524

(74) 대리인
전영일
임주석

심사청구 : 없음

(54) 프로그램가능한 회로 및 관련 컴퓨팅 머신 및 방법

요약

프로그램가능한 회로는 회부 소스로부터 구성 데이터를 수신하고, 메모리에 펌웨어를 저장하고, 상기 펌웨어를 상기 메모리로부터 다운로드 한다. 이러한 프로그램가능한 회로는 컴퓨팅 머신 등의 시스템으로 하여금 상기 프로그램가능한 회로의 구성을 수정할 수 있게 하여 구성 메모리를 수동으로 재프로그래밍하는 필요를 없애준다. 예를 들어, 만일 상기 프로그램가능한 회로가 파이프라인 가속기의 일부인 FPGA라면, 상기 가속기와 결합되어 있는 프로세서는 FPGA의 구성을 수정할 수 있다. 보다 특별하게는, 프로세서는 구성 레지스트리로부터 상기 수정된 구성을 나타내는 펌웨어를 검색하고, 그 펌웨어를 FPGA로 전송한 다음, 이 펌웨어를 전기적으로 지울 수 있고 프로그램가능한 판독-전용 메모리(EEPROM)와 같은 메모리에 저장한다. 다음으로, FPGA 는 상기 펌웨어를 상기 메모리로부터 그의 구성 레지스트리로 다운로드하여 상기 수정된 구성을 갖도록 자기 자신을 재구성한다.

대표도

도 3

색인어

프로그램가능한 회로, FPGA

명세서

기술분야

(우선권 주장)

본 출원은 참고문헌으로서 통합되는 2002년 10월 31일 출원된 미국 가출원 제60/422,503호의 우선권을 주장한다.

(관련된 출원과의 상호참조)

본 출원은 발명의 명칭이 "향상된 컴퓨팅 아키텍처 및 관련 시스템 및 방법" 인 미국 특허출원 제10/684,102호, 발명의 명칭이 "향상된 컴퓨팅 아키텍처를 가지는 컴퓨팅 머신 및 관련 시스템 및 방법" 인 미국 특허출원 제 10/684,053호, 발명의 명칭이 "향상된 컴퓨팅 아키텍처를 위한 파이프라인 가속기 및 관련 시스템 및 방법" 인 미국 특허출원 제10/683,929호, 및 발명의 명칭이 "다중 파이프라인 유닛을 가지는 파이프라인 가속기 및 관련 컴퓨팅 머신 및 방법" 인 미국 특허출원 제 10/684,932호와 관련이 되어 있으며, 상기 미국 특허출원 발명은 모두 2003년 10월 9일 동일한 권리자에 의해 출원되었으며, 본 명세서에 참고문헌으로 통합된다.

배경기술

상대적으로 짧은 시간에서 상대적으로 대용량의 데이터를 처리하기 위한 일반적인 컴퓨팅 아키텍처(computing architecture)에는 부하를 분배하는 다중 상호접속 프로세서(multiple interconnected processor)가 포함되어 있다. 처리 부하를 분배함으로써, 이들 다중 프로세서들은 주어진 클록 주파수에서 하나의 프로세서가 할 수 있는 것 보다 더욱 빨리 데이터를 처리할 수 있다. 예를 들어, 프로세서 각각은 데이터의 일부를 처리 하거나 또는 처리되는 알고리즘 일부를 실행할 수 있다.

도 1은 다중-프로세서 아키텍처를 갖는 종래의 컴퓨팅 머신(computing machine)(10)의 개략적인 블록 다이어그램이다. 머신(10)에는 마스터 프로세서(12) 및 버스(16)를 통해 상기 마스터 프로세서와 상호 통신을 하는 코프로세서(14_1-14_n), 원격 장치(도 1에는 도시하지 않음)로부터 원 데이터(raw data)를 수신하는 입력 포트(18), 및 처리된 데이터를 상기 원격 소스로 제공하는 출력 포트(20)가 포함되어 있다. 상기 머신(10)에는 또한 마스터 프로세서(12)용 메모리(22), 코프로세서(14_1-14_n)용 메모리(24_1-24_n), 및 상기 버스(16)를 통해 마스터 프로세서와 코프로세서가 공유하는 메모리(26)도 포함되어 있다. 메모리(22)는 마스터 프로세서(12)를 위한 프로그램 및 작업 메모리 모두의 역할을 하고, 메모리(24_1-24_n) 각각은 코프로세서(14_1-14_n) 각각을 위한 프로그램 및 작업 메모리 모두의 역할을 한다. 공유 메모리(26)는 마스터 프로세서(12)와 코프로세서(14)가 그들 사이의 데이터를 포트(18 및 20)을 통해 각각 원격 장치로/장치로부터 전달할 수 있게 해준다. 마스터 프로세서(12) 및 코프로세서(14)는 또한 머신(10)이 원 데이터를 처리하는 속도를 제어하는 공통의 클록 신호를 수신하기도 한다.

일반적으로, 컴퓨팅 머신(10)은 마스터 프로세서(12)와 코프로세서(14) 간의 원 데이터 처리를 효과적으로 분배한다. 소나 어레이(sonar array)과 같은 원격 소스(도 1에는 도시하지 않음)는 포트(18)를 통해 원 데이터를 상기 원 데이터를 위한 선입선출(FIFO) 버퍼(도시하지 않음)로서 작동하는 공유 메모리(26)의 일부에 로드한다. 마스터 프로세서(12)는 버스(16)를 통해 메모리(10)로부터 원 데이터를 검색하고, 마스터 프로세서와 코프로세서(14)가 상기 원 데이터를 처리하고, 버스(16)를 통해 필요한 만큼 그들사이에서 데이터를 전달한다. 마스터 프로세서(12)는 처리된 데이터를 공유 메모리(26)에 정의되어 있는 다른 FIFO 버퍼(도시하지 않음)에 로드하고, 원격 소스가 포트(20)를 통해 이 FIFO 로부터 상기 처리된 데이터를 검색한다.

다른 연산의 예에서, 컴퓨팅 머신(10)은 원 데이터를 처리하는데 있어서 상기 원 데이터상에서 각각의 연산에 n+1을 순차적으로 수행하여 처리하는데, 이 연산은 패스트 푸리에 변환(FFT)과 같은 처리 알고리즘을 함께 구성한다. 보다 특별하게는, 머신(10)은 마스터 프로세서(12)와 코프로세서(14)로부터 데이터-처리 파이프라인을 형성한다. 주어진 클록 신호의 주파수를 위해, 그러한 파이프라인은 종종 머신(10)이 하나의 프로세서만 가지는 머신보다 더욱 빠르게 원 데이터를 처리할 수 있게 한다.

메모리(26)내의 원-데이터 FIFO(도시하지 않음)로부터 원 데이터를 검색한 후, 마스터 프로세서(12)는 삼각 함수와 같은 제1 연산을 상기 원 데이터상에 수행한다. 이 연산은 프로세서(12)가 메모리(26) 내부에 정의된 제1-결과 FIFO(도시하지 않음)내에 저장하는 첫번째 결과를 산출해 낸다. 일반적으로, 프로세서(12)는 메모리(12) 내에 저장된 프로그램을 수행하며, 그 프로그램의 제어하에 상기 설명된 연산등을 수행한다. 프로세서(12)는 또한 메모리(22)를 작업 메모리로 사용하여 프로세서가 상기 제1 연산의 중간 시간에 발생하는 데이터를 일시적으로 저장하기도 한다.

다음으로, 상기 메모리(26)내의 제1-결과 FIFO(도시하지 않음)로부터 첫번째 결과를 검색한 후, 코프로세서(14)는 로그 함수와 같은 두번째 연산을 상기 첫번째 결과상에 수행한다. 이 두번째 연산은 코프로세서(14_1)가 메모리(26) 내부에 정의된 제2-결과 FIFO(도시하지 않음)내에 저장하는 두번째 결과를 산출해 낸다. 일반적으로, 코프로세서(14_1)는 메모리(24_1) 내에 저장된 프로그램을 수행하며, 그 프로그램의 제어하에 상기 설명된 연산등을 수행한다. 코프로세서(14)는 또한 메모리(24_1)를 작업 메모리로 사용하여 코프로세서가 상기 제2 연산의 중간 시간에 발생하는 데이터를 일시적으로 저장하기도 한다.

그리고 나서, 코프로세서(24_2-24_n)는 상기 두번째-(n-1)번째 상에 세번째-n번째 연산을 순차적으로 수행하여 상기 코프로세서(24_1)를 위해 상기 설명한 것과 유사한 방법의 결과를 가져온다.

코프로세서(24_n)에 의해 수행되는 n번째 연산은 최종 결과, 즉 처리된 데이터를 산출한다. 코프로세서(24_n)는 메모리(26) 내부에 정의된 처리된-데이터 FIFO(도시하지 않음)로 이 처리된 데이터를 로드(load)하고, 원격 장치(도 1에는 도시하지 않음)가 이 FIFO로부터 상기 처리된 데이터를 검색한다.

마스터 프로세서(12)와 코프로세서(14)가 처리 알고리즘의 다른 연산을 동시에 수행하기 때문에, 컴퓨팅 머신(10)은 서로 다른 연산을 순차적으로 수행하는 하나의 프로세서를 갖는 컴퓨팅 머신 보다도 원 데이터를 보다 빨리 처리할 수 있기도 하다. 특히, 하나의 프로세서는 원 데이터의 앞 세트상에 모든 n+1 연산을 수행할 때 까지는 원 데이터의 새로운 세트를 검색할 수 없다. 그러나, 상기 언급한 파이프라인 기술을 이용해서, 마스터 프로세서(12)는 오직 첫번째 연산을 수행한 후 원 데이터의 새로운 세트를 검색할 수 있다. 따라서, 주어진 클록 주파수를 위해, 이 파이프라인 기술은 머신(10)이 원 데이터를 처리하는데 있어서 하나의 프로세서 머신(도 1에는 도시하지 않음)과 비교할 때 대략 n+1 팩터 만큼 원 데이터를 처리하는 속도를 증가시킬 수 있다.

선택적으로, 컴퓨팅 머신(10)은 원 데이터상에, FFT 와 같은 처리 알고리즘의 경우에 n+1을 동시에 수행함으로써 병렬로 원 데이터를 처리할 수도 있다. 즉, 알고리즘에 앞서의 실시예에서 상기 언급한 바와 같은 n+1 순차적 연산이 포함되어 있다면, 마스터 프로세서(12)와 코프로세서(14) 각각은 모든 n+1 연산을 원 데이터 각각의 세트상에서 수행한다. 따라서, 주어진 클록 주파수를 위해서는, 상기 설명한 파이프라인 기술과 같이, 이러한 병렬-처리 기술은 머신(10)이 하나의 프로세서 머신(도 1에는 도시하지 않음)과 비교할 때 대략 n+1 팩터 만큼 원 데이터 처리 속도를 증가시킬 수 있다.

불행하게도, 비록 컴퓨팅 머신(10)이 하나의 프로세서 컴퓨팅 머신(도 1에는 도시하지 않음)보다 빨리 데이터를 처리할 수는 있으나, 머신(10)의 데이터-처리 속도가 종종 프로세서 클록의 주파수보다 적어지곤 한다. 특히, 컴퓨팅 머신(10)의 데이터-처리 속도는 마스터 프로세서(12)와 코프로세서(14)가 데이터를 처리하는데 요구하는 시간에 의해 제한된다. 간략히 하기 위해, 이 속도 제한의 한 예를 마스터 프로세서(12)를 참고하여 설명하는데, 이 설명은 코프로세서(14)에도 적용됨을 이해할 수 있을 것이다. 앞에서 설명한 바와 같이, 마스터 프로세서(12)는 프로세서를 제어하여 데이터를 원하는 방식으로 조작하도록 제어하는 프로그램을 실행한다. 이 프로그램에는 프로세서(12)가 실행하는 일련의 명령이 포함되어 있다. 불행하게도, 프로세서(12)는 일반적으로 하나의 명령을 수행하는데 다중 클록 사이클을 필요로 하는데, 종종 다중 명령을 수행하여 데이터의 하나의 값을 처리해야 한다. 예를 들어, 프로세서(12)가 제1 데이터 값(A)(도시하지 않음)과 제2 데이터 값(B)(도시하지 않음)을 곱하는 경우를 가정한다. 제1 클록 사이클 동안, 프로세서(12)는 메모리(22)로부터 여러개의 명령을 검색한다. 제2 및 제3 클록 사이클 동안, 프로세서(12)는 메모리(22)로부터 A와 B를 각각 검색한다. 제4 클록 사이클 동안, 프로세서(12)는 A와 B를 곱하고, 제5 클록 사이클 동안, 그 결과물을 메모리(22 또는 26)에 저장하거나 또는

그 결과물을 원격 장치(도시하지 않음)로 제공한다. 이것은 최선의 시나리오인데, 그 이유는 많은 경우에서, 프로세서(12)는 카운터를 초기화(initializing) 및 닫는(closing) 경우와 같이 오버헤드 태스크(overhead task)를 위한 추가의 클록 사이클을 요구하기 때문이다. 그러므로, 프로세서(12)는 A와 B를 처리하기 위해서는 5개의 클록 사이클, 또는 데이터 값 당 평균 2.5개의 클록 사이클을 필요로 한다.

따라서, 컴퓨팅 머신(10)이 데이터를 처리하는 속도는 종종 마스터 프로세서(12) 및 코프로세서(14)를 구동하는 클록의 주파수보다 크게 낮아지곤 한다. 예를 들어, 프로세서(12)가 1.0 기가헤르쯔(GHz)에서 클록되지만 데이터 값 당 평균 2.5 클록 사이클을 요구한다면, 유효 데이터-처리 속도는 (1.0GHz)/2.5=4GHz 가 될 것이다. 이 유효 데이터-처리 속도는 종종 초당 연산 단위로 특징되곤 한다. 그러므로, 1.0GHz 의 클록 속도를 위해서는, 프로세서(12)는 초당 0.4 기가연산(Gops)의 데이터-처리 속도로 레이트(rate)되어야 한다.

도 2는 프로세서가 주어진 클록 주파수 및 종종 파이프라인이 클록되는 레이트와 거의 동일한 레이트에서 할 수 있는 것보다 더 빠른 일반적인 데이터를 처리할 수 있는 하드와이어드(hardwired) 데이터 파이프라인(30)의 블록 다이어그램이다. 파이프라인(30)에는 각각 실행 프로그램 명령 없이 각각의 데이터상의 개별적 연산을 각각 수행하는 연산자 회로(32_1-32_n)가 조합되어 있다. 즉, 원하는 연산이 회로(32)로 "번 되는(burned in)" 것으로 프로그램 명령 없이도 자동적으로 연산을 실행하는 것이다. 실행 프로그램 명령과 관련된 오버헤드를 제어함으로써, 파이프라인(30)은 종종 주어진 클록 주파수를 위해 할 수 있는 것보다 더 많은 연산을 수행할 수 있다.

예를 들어, 파이프라인(30)은 프로세서가 주어진 클록 주파수를 위해 할 수 있는 것보다 더 빠른 다음과 같은 수식을 풀 수 있다.

$$Y(X_k)=(5X_k+3)2^{Xk}$$

여기서, Xk는 입력의 원 데이터 값을 나타낸다. 이 예에서, 연산자 회로(32_1)는 5Xk 를 계산하는 곱셈기이고, 회로(32_2)는 5Xk + 3 을 계산하는 가산기이며, 회로(32_n)(n=3)는 (5Xk + 3)2^{Xk} 를 계산하는 곱셈기이다.

제1 클록 사이클 k=1 동안, 회로(32_1)는 데이터 값(X_1)을 수신하고 여기에 5를 곱해 $5X_1$ 을 발생한다.

제2 클록 사이클 k=2 동안, 회로(32_2)는 회로(32_1)로부터 $5X_1$을 수신하고, 3을 더해서 $5X_1$ + 3 을 발생한다. 또한, 상기 제2 클록 사이클 동안, 회로(32_1)는 $5X_2$ 를 발생한다.

제3 클록 사이클 k=3 동안, 회로(32_3)는 회로(32_2)로부터 $5X_1$ + 3 을 수신하고, 2^{X1} (x1 만큼 유효하게 $5X_1$ + 3 왼쪽 시프트)를 곱하여 첫번째 결과($5X_1$ + 3)2^{X1} 을 발생한다. 또한, 제3 클록 사이클 동안, 회로(32_1)는 $5X_3$ 를 발생하고 회로(32_2)는 $5X_2$ + 3 을 발생한다.

파이프라인(30)은 이러한 방식으로 모든 원 데이터 값이 처리될 때 까지 연속적인 원 데이터 값(X_k)을 계속 처리한다.

따라서, 원 데이터 값(X_1)을 수신한 후 두 개의 클록 사이클의 지연 - 이 지연을 파이프라인(30)의 레이턴시(latency)라고 부르곤 함 - 상기 파이프라인은 결과 ($5X_1$ + 3)2^{X1} 을 발생하고, 그 후에 하나의 결과 - 예를 들어, ($5X_2$+ 3)2^{X2}, ($5X_3$+ 3)2^{X3},...$5X_n$+ 3)2^{Xn}, 를 각각의 클록 사이클을 발생한다.

상기 레이턴시를 무시하면, 파이프라인(30)은 클록 속도와 동일한 데이터-처리 속도를 갖는다. 비교를 하면, 마스터 프로세서(12)와 코프로세서(14)가 상기 예에서와 같은 클록 속도의 0.4배의 데이터-처리 속도를 갖는다고 가정하면, 파이프라인(30)은 주어진 클록 속도를 위해 컴퓨팅 머신(10)(도 1)보다 2.5배 빨리 데이터를 처리할 수 있다.

도 2를 계속 참조하면, 설계자는 파이프라인(30)을 필드-프로그램가능한 게이트 어레이(FPGA)와 같은 프로그램가능한 로직 IC(PLIC)에서 수행하도록 선택하기도 하는데, 그 이유는 PLIC 는 주문형 반도체(ASIC)보다 나은 디자인 및 변형 유연성 가진다. PLIC 내부에서 하드와이어드 접속을 구성하기 위해서는, 설계자는 단지 PLIC 내부에 배치된 상호접속-구성 레지스터를 미리 결정된 이진 상태(binary state)로 설정하기만 하면 된다. 이들 이진 상태 모두의 조합을 "펌웨어(firmware)"라고 부르곤 한다. 일반적으로, 설계자는 이 펌웨어를 PLIC 와 결합된 비휘발성 메모리(도 2에 도시하지 않음)에 프로한다. 누군가가 PLIC 를 "켜면(turn on)", PLIC는 상기 메모리로부터 펌웨어를 상기 상호접속-구조 레지스터로 다운로드한다. 그러므로, PLIC 의 기능을 변경시키기 위해서는, 설계자는 단지 펌웨어를 수정하면 되고 PLIC 로 하여금 그 수정된 펌웨어를 상호접속-구조 레지스터로 다운로드하게 하면 된다. 이것은 단지 펌웨어를 수정하는 것에 의해 PLIC를 수정할 수 있다는 것은 시제품화 단계 동안 및 "필드 내에서" 파이프라인(30)의 업그레이드를 위해 특히 유용하다.

불행하게도, 하드와이어드 파이프라인(30)은 중요한 결정 형성, 특히 내포된 결정 형성(nested decision making)을 필요로 하는 알고리즘을 실행하는데 최선의 선택이 되지 못한다. 프로세서는 비교가능한 길이의 연산 명령(예를 들어, "A+B")를 실행할 수 있는 정도로 거의 빠르게 일반적으로 내포된-결정-형성 명령(예를 들어, "A이면 B를 행하고, 그렇지 않고 C이면 D를 행하고, ... 그렇지 않으면 n" 과 같은 내포된 조건 명령)를 실행할 수 있다. 그러나, 비록 파이프라인(30)이 상대적으로 간단한 결정(예를 들어, "A>B?")을 유효하게 구성할 수 있어도, 일반적으로 내포된 결정(예를 들어, "A이면 B를 행하고, 그렇지 않고 C이면 D를 행하고, ... 그렇지 않고 n")을 프로세서가 할 수 있는 것 만큼 유효하게 실행할 수 없다. 이러한 비효율성의 한 이유는 파이프라인(30)은 온-보드 메모리(on-board memory)가 거의 없어서 외부 작업/프로그램 메모리(도시하지 않음)를 액세스 할 필요가 있기 때문이다. 그리고, 비록 그러한 내포된 결정을 실행하기 위해 파이프라인(30)을 디자인할 수 있다 하여도, 요구되는 회로의 크기와 복잡성은 종종 설계를 불가능하게 만드는데, 특히 여러개의 서로 다른 내포된 결정을 포함하는 알고리즘인 경우 그러하다.

따라서, 프로세서는 중요한 결정 형성을 요구하는 애플리케이션 내에서 종종 사용되며, 하드와이어드 파이프라인은 결정 형성이 거의 없는 또는 전혀 없는 "수치처리(number crunching)" 애플리케이션으로 제한되곤 한다.

더욱이, 아래에 설명한 바와 같이, 도 2의 파이프라인(30)과 같은 하드와이어드 파이프라인, 특히 여러개의 PLIC를 포함하는 파이프라인(30)을 설계/수정하는 것 보다는 도 1의 컴퓨팅 머신(10)과 같은 프로세서-기반 컴퓨팅 머신을 설계/수정하는 것이 훨씬 쉽다.

프로세서과 그 주변장치등(예를 들어, 메모리)과 같은 컴퓨팅 구성요소들은 일반적으로 이 구성요소들의 상호접속을 촉진하여 프로세서-기반 컴퓨팅 머신을 형성하기 위한 산업-표준 통신 인터페이스를 포함한다.

특히, 표준 통신 인터페이스에는 두 개의 계층(layer)이 포함되는데, 물리 계층과 서비스 계층이다.

물리 계층에는 회로 및 이 회로의 연산 파라메터 및 통신 인터페이스를 형성하는 대응 회로 상호접속이 포함되어 있다. 예를 들어, 물리 계층에는 구성요소들 버스와 인결하는 핀, 이 핀으로부터 데이터를 래치(latch)하는 버퍼, 이 핀상에서 신호를 구동시키는 구동기, 및 입력 데이터 신호로부터 데이터를 복구하고 상기 데이터 신호 또는 외부 물독 신호로부터 물독 신호를 복구하는 회로가 포함되어 있다. 상기 연산 파라메터에는 상기 핀이 수신하는 데이터 신호의 허용가능한 전압 범위, 데이터를 기록 및 판독하는 신호 타이밍, 및 지원된 연산 모드(예를 들어, 버스트 모드, 페이지 모드)가 포함되어 있다. 종래의 물리 계층에는 트랜지스터-트랜지스터 논리(TTL) 및 램버스(RAMBUS)가 포함된다.

서비스 계층에는 컴퓨팅 구성요소가 데이터를 전송하는 프로토콜이 포함된다. 이 프로토콜은 데이터의 포맷 및 상기 구성요소가 포맷된 데이터를 송수신하는 방식을 정의한다. 종래의 통신 프로토콜에는 파일-전송 프로토콜(FTP) 및 전송 제어 프로토콜/인터넷 프로토콜(TTP/IP)이 포함된다.

따라서, 산업-표준 통신 인터페이스를 갖는 제조자 및 다른 일반적인 설계 컴퓨팅 구성요소들로 인해서, 그러한 구성요소의 인터페이스를 일반적인 설계로 할 수 있고 이것을 상대적으로 적은 노력으로 다른 컴퓨팅 구성요소들과 상호접속시킬 수 있다. 이것은 설계자에게 대부분의 시간을 컴퓨팅 머신의 다른 부분들을 설계하는데 소비하게 만들고, 구성요소들을 추가하거나 없애는 것을 통해 머신을 쉽게 수정할 수 있게 한다.

산업-표준 통신 인터페이스를 지원하는 컴퓨팅 구성요소를 설계하는 것은 설계 라이브러리(design library)로부터 현존하는 물리-계층 설계를 사용하여 설계 시간을 절약하게 해 준다. 이것은 또한 구성요소들은 제고품인 컴퓨팅 구성요소들과 쉽게 접속할 수 있게 해주기도 한다.

공통의 산업-표준 통신 인터페이스를 지원하는 컴퓨팅 구성요소를 사용하여 컴퓨팅 머신을 설계하는 것은 설계자로 하여금 시간과 노력을 줄여주면서 구성요소들을 상호접속할 수 있게 해 준다. 이들 구성요소들이 공통의 인터페이스를 지원하기 때문에, 설계자는 설계 노력을 거의 들이지 않고 시스템 버스를 통해 이들을 상호 접속할 수 있다. 지원되는 인터페이스가 산업 표준이기 때문에, 설계자는 머신을 쉽게 수정할 수 있다. 예를 들어, 설계자는 다른 구성요소 및 주변장치들을 머신에 추가하여 시스템 설계를 발전시켜 나갈 수 있으며, 또는 차세대 구성요소들을 쉽게 추가/설계하여 기술 발전을 이룰 수 있다. 더욱이, 구성요소들이 공통의 산업-표준 서비스 계층을 지원하기 때문에, 설계자는 컴퓨팅 머신의 소프트웨어로 대응하는 프로토콜을 실현하는 한존하는 소프트웨어 모듈을 합체시킬 수 있다. 따라서, 설계자는 인터페이스 설계가 이미 적절하게 필수적이기 때문에 거의 노력을 들이지 않고 구성요소들을 접속시킬 수 있어서 머신이 특정 기능을 수행하게 하는 머신의 일부(예를 들어, 소프트웨어)를 설계하는 데 주력할 수 있다.

그러나, 불행하게도, 도 2의 파이프라인(30)과 같은 하드와이어드 파이프라인을 형성하는데 사용되는 PLIC 등과 같은 구성요소를 위한 알려진 산업-표준 서비스 계층은 없다.

따라서, 여러개의 PLIC 를 갖는 파이프라인을 설계하기 위해서, 설계자는 "스크래치(scratch)로부터" 설계를 하고 PLIC 간의 통신 인터페이스의 서비스 계층을 디버깅하는데 상당한 시간과 노력을 들여야 한다. 일반적으로, ad hoc 서비스 계층은 PLIC 간에서 전달되는 데이터의 파라메터에 따라 달라진다. 비슷하게, 프로세서와 접속되는 파이프라인을 설계하기 위해서는, 설계자는 파이프라인과 프로세서간의 통신 인터페이스의 서비스 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다.

비슷하게, PLIC 을 추가하는 것으로 파이프라인을 수정하기 위해서는, 설계자는 일반적으로 추가된 PLIC와 현재의 PLIC 사이의 통신 인터페이스의 서비스 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들이게 된다. 그리고, 프로세서를 추가하는 것으로 파이프라인을 수정하려면, 또는, 파이프라인을 추가하는 것으로 컴퓨팅 머신을 수정하려면, 설계자는 파이프라인과 프로세서간의 통신 인터페이스의 서비스 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다.

그러므로, 도 1 및 도 2를 참고하면, 다수의 PLIC 를 접속하고 파이프라인과 프로세서의 접속 어려움으로 인해, 설계자는 컴퓨팅 머신을 설계하는 경우 상당한 트레이드오프(tradeoff)에 직면하곤 한다. 예를 들어, 프로세서-기반 컴퓨팅 머신을 가지고는, 설계자는 복잡한 결정-형성 가능성을 위한 트레이드 수-코딩칭 속도 및 설계/수정 유연성에 집중하게 된다. 반대로, 하드와이어드 파이프라인-기반 컴퓨팅 머신을 가지고는, 설계자는 수-코딩칭 속도를 위한 트레이드 복잡성-결정-형성 가능성 및 설계/수정에 집중하게 된다. 더욱이, 다수의 PLIC 를 접속하는데의 어려움으로 인해, 소수의 PLIC 를 가지는 파이프라인-기반 머신을 설계하는 것이 불가능하기도 하다. 그 결과, 실제적인 파이프라인-기반 머신은 제한된 기능을 갖곤 한다. 그리고, 프로세서와 PLIC 와의 접속 어려움으로 인해서, 하나의 PLIC 이상과 프로세서와의 접속이 불가능하기도 하다. 그 결과, 프로세서와 파이프라인을 합침으로써 얻어지는 이익이 적다.

따라서, 하드와이어드-파이프라인-기반 머신의 수-코딩칭 속도를 가지고 프로세서-기반 머신의 결정-형성 가능성을 결합시킬 수 있는 새로운 컴퓨팅 아키텍쳐 요구가 있어 왔다.

발명의 상세한 설명

(요약)

본 발명의 일 실시예에 따르면, 프로그램가능한 회로가 외부 소스로부터 펌웨어를 수신하고, 상기 펌웨어를 메모리에 저장하고, 그리고 상기 펌웨어를 상기 메모리로부터 다운로드한다.

그러한 프로그램가능한 회로는 컴퓨팅 머신과 같은 시스템으로 하여금 프로그램가능한 회로의 구성을 수정할 수 있게 하여 구성 메모리를 수동으로 재프로그래밍 하는 필요를 없애준다. 예를 들어, 프로그램가능한 회로가 파이프라인 가속기의 일부인 FPGA 인 경우, 상기 가속기와 결합된 프로세서는 상기 FPGA 의 구성을 수정할 수 있다. 보다 특별하게는, 상기 프로세서는 구성 레지스트리로부터 상기 수정된 구성을 나타내는 펌웨어를 검색하고 그 펌웨어를 FPGA 로 전송하는데, 상기 펌웨어는 전기적으로 지울수 있는 판독 전용 메모리(EEPROM)과 같은 메모리에 저장된다. 다음으로, FPGA는 상기 펌웨어를 상기 메모리로부터 그의 구성 레지스터로 다운로드하여 자기 자신이 상기 수정된 구성을 갖도록 효과적으로 재구성한다.

도면의 간단한 설명

도 1은 종래의 다중-프로세서 아키텍처를 갖는 컴퓨팅 머신의 블럭 다이어그램이고,

도 2는 종래의 하드와이어드 파이프라인의 블럭 다이어그램이고,

도 3은 본 발명의 일 실시예에 따른 피어-벡터 아키텍처를 갖는 컴퓨팅 머신의 블럭 다이어그램이고,

도 4는 본 발명의 일 실시예에 따른 도 3의 파이프라인 가속기의 파이프라인 유닛의 블럭 다이어그램이고,

도 5는 본 발명의 일 실시예에 따른 도 4의 펌웨어 메모리의 논리적 파티셔닝의 다이어그램이고,

도 6은 본 발명의 더 다른 실시예에 따른 도 3의 파이프라인 가속기의 파이프라인 유닛의 블럭 다이어그램이다.

실시예

(상세한 설명)

도 3은 본 발명의 일 실시예에 따른 피어-벡터 아키텍처를 갖는 컴퓨팅 머신(40)의 개략적인 블럭 다이어그램이다. 호스트 프로세서(42)에 추가하여, 상기 피어-벡터 머신(40)에는 적어도 일부의 데이터 처리를 수행하여 도 1의 컴퓨팅 머신(10) 내의 코프로세서(14)의 뱅크(bank)를 유효하게 대체하는 파이프라인 가속기(44)가 포함되어 있다. 따라서, 호스트-프로세서(42) 및 가속기(44)(또는 후술할 파이프라인 유닛)는 데이터 패킷을 앞뒤로 전송할 수 있는 "피어(peer)" 이다. 가속기(44)는 프로그램 명령을 실행하지 않으므로, 가속기는 종종 코프로세서의 뱅크가 주어진 클록 주파수에서 할 수 있는 것보다 훨씬 빠르게 데이터상의 집중적인 연산을 수학적으로 처리한다. 따라서, 프로세서(42)의 결정-형성 가능성과 가속기(44)의 수-크런칭 가능성을 결합함으로써, 머신(40)은 동일한 능력을 갖지만, 머신(10)과 같은 종래의 컴퓨팅 머신보다는 빠르게 데이터를 처리할 수 있다. 또한, 후술하는 바와 같이, 가속기(44)에 상기 호스트 프로세서(42)의 통신 인터페이스와 호환되는 통신 인터페이스를 제공하는 것은 머신(40)의 설계 및 수정을 용이하게 해 주고, 특히, 프로세서의 통신 인터페이스가 산업 표준인 경우 그러하다. 상기 가속기(44)는 하나 또는 그 이상의 PLIC가 포함되어 있고, 호스트 프로세서(42)는 적절한 펌웨어를 이들 PLIC로 전송함으로써 가속기 내부의 물리적인 상호접속을 하드 구성할 수 있다. 호스트 프로세서(42)는 피어-벡터 머신(40)의 초기화 동안 이 방식으로 가속기(44)를 구성을 할 수 없지만, 아래에서 설명하고 앞서 인급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,053호에 설명된 바와 같이 피어-벡터 머신의 동작 동안 상기 가속기를 재구성할 수 있다. 더욱이, 피어-벡터 머신(40)은 후술하는 바와 같은 그리고 앞서 인급한 다른 출원들에서의 다른 장점도 제공한다.

도 3을 계속 참조하면, 호스트 프로세서(42) 및 파이프라인 가속기(44)에 추가하여, 피어-벡터 컴퓨팅 머신(40)에는 프로세서 메모리(46), 인터페이스 메모리(48), 파이프라인 버스(50), 하나 또는 그 이상의 메모리(52), 선택적인 원-데이터 입력 포트(54), 처리된-데이터 출력 포트(58), 선택적인 라우터(61) 및 테스트 버스(63)가 포함되어 있다.

호스트 프로세서(42)에는 처리 유닛(62) 및 메시지 처리기(64)가 포함되어 있으며, 프로세서 메모리(46)에는 처리-유닛 메모리(66) 및 처리기 메모리(68)을 포함하는데, 각각 프로세서 유닛 및 메시지 처리기를 위한 프로그램 및 작업 메모리 모두의 역할을 한다. 프로세서 메모리(46)에는 가속기-구성 레지스트리(70) 및 메시지-구성 레지스트리(72)도 포함되어 있는데, 이들은 각각 호스트 프로세서(42)로 하여금 가속기(44)의 기능 및 메시지 처리기(64)가 송수신하는 메시지의 포맷을 구성하도록 하는 펌웨어 및 구성 데이터를 저장하고 있다.

가속기(44) 및 메시지 처리기(64)의 구성은 앞서 인급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호에 더 설명되어 있고, 가속기(44)의 구성은 도 6을 참조하여 아래에 더 설명한다.

파이프라인 가속기(44)는 적어도 하나의 PLIC(도 4)에 배치되어 있고 프로그램 명령을 실행하지 않고 각각의 데이터를 처리하는 하드와이어드 파이프라인(74_1-74_n)을 포함하고 있다. 상기 펌웨어 메모리(74)는 가속기(44)를 위해 상기 펌웨어를 저장한다. 보다 특별하게는, 상기 펌웨어 메모리(74)는 도 4 내지 도 6을 참조하여 아래에 더 설명하는 바와 같이 가속기(44)를 포함하는 PLIC를 위해 상기 펌웨어를 저장한다. 전형적으로, 상기 가속기(44)는 적어도 하나의 ASIC 에 배치될 수 있으며, 따라서, 일단 ASIC가 형성되면 구성할 수 없는 내부 상호접속을 갖는다. 가속기(44)에 PLIC 가 포함되지 않

는 이 대안에서, 머신(40)에서 펌웨어 메모리(52)를 생략할 수 있다. 또한, 비록 가속기(44)가 다중 파이프라인(74_1-74_n)을 포함하는 것으로 도시되어 있으나, 오직 하나의 파이프라인만 포함할 수 있다. 또한, 비록 도시하지는 않았지만, 가속기(44)에는 디지털-신호 처리기(DSP)와 같은 하나 또는 그 이상의 프로세서가 포함될 수 있다. 또한, 비록 도시하지는 않았지만, 가속기(44)에는 데이터 입력 포트 및/또는 데이터 출력 포트를 포함할 수 있다.

파이-빅터 머신(40)의 일반적인 동작은 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,102호에 설명되어 있고, 호스트 프로세서(42)의 구조 및 동작은 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호에 설명되어 있으며, 파이프라인 가속기(44)의 구조 및 동작은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호 및 발명의 명칭이 "PIPELINE ACCELERATOR HAVING MULTIPLE PIPELINE UNITS AND RELATED COMPUTING MACHINE AND METHOD" 인 미국 특허출원 제10/683,932호에 설명되어 있다. 가속기(44)를 구성하는 PLIC 의 동작 구성은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 설명되어 있으며, 아래 도 4 내지 도 6을 참조하여 설명한다.

도 4 내지 도 6을 참고하여, 가속기(44) PLIC를 "하드" 구성하기 위한 기술을 설명한다. 앞서 설명한 바와 같이, PLIC의 하드 구성은 펌웨어에 의해 프로그램되며 PLIC 의 구성요소들간의 특정한 물리적 상호접속을 나타내는데, 즉, 하나의 로직 블록이 어떻게 다른 로직 블록과 전기적으로 연결되는가 하는 것이다. 이것은 이미 하드-구성된 PLIC의 상위-레벨 구성을 나타내는 "소프트" 구성과는 다르다. 예를 들어, 하드-구성된 PLIC에는 비교가 포함되기도 하며, 어느 하나를 소프트 구조가 되게 하여 대응하는 소프트-구성 데이터를 레지스터에 로드함으로써 상기 비교의 크기를 구성하게 하는 레지스터가 포함되기도 한다. 가속기(44)의 소프트 구성은 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호 및 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

도 4는 본 발명의 일 실시예에 따른 도 3의 가속기(44)의 파이프라인 유닛(74)의 블록 다이어그램이다. 하드와이어드 파이프라인(74_1-74_n)(도 3)은 후술하는 바와 같이 하드와이어드 파이프라인을 제어하고 이들로 하여금 데이터를 수신하고 전송하고 저장하게 하는 회로를 포함하는 파이프라인 유닛(78)의 일부이다. 도 4에는 비록 하나의 파이프라인(78)만이 도시되어 있으나, 가속기(44)에는 앞서 언급한 바와 같이 발명의 명칭이 "PIPELINE ACCELERATOR HAVING MULTIPLE PIPELINE UNITS AND RELATED COMPUTING MACHINE AND METHOD" 인 미국 특허출원 제10/683,932호에 설명된 바와 같은 다수의 파이프라인 유닛(각각에는 상기 하드와이어드 파이프라인(74_1-74_n) 중 적어도 일부가 포함된다)이 포함될 수 있다. 후술하는 바와 같이, 한 수행에서, 파이프라인 유닛(78)의 하드 구성은 펌웨어로 프로그램가능하다. 이것은 설계자로 하여금 단지 펌웨어만 수정함으로써 파이프라인 유닛(78)의 기능을 수정할 수 있게 해준다. 또한, 호스트 프로세서(42)(도 3)은 파이-빅터 머신(40)(도 3)의 초기화 또는 재구성 동안에 상기 수정된 펌웨어를 파이프라인 유닛(78)으로 제공할 수 있어서, 설계자가 상기 수정된 펌웨어를 파이프라인 유닛으로 수동으로 로드할 필요를 없게 해준다.

파이프라인 유닛(78)에는 PLIC 또는 ASIC와 같은 파이프라인 회로(80), 펌웨어 메모리(52)(여기서 상기 파이프라인 회로는 PLIC 임), 및 데이터 메모리(81)가 포함되는데, 이들은 모두 회로 보드 또는 카드(83)상에 배치된다. 데이터 메모리(81)는 앞서 언급한 발명의 명칭이 "PROGRAMMABLE CIRCUIT AND RELATED COMPUTING MACHINE AND METHOD"인 미국 특허출원 제10/684,057호에 더 설명되어 있으며, 파이프라인 회로(80)와 펌웨어 메모리(52)의 결합은 프로그램가능한 회로 유닛을 형성한다.

상기 파이프라인 유닛(80)에는 호스트 프로세서(42)(도 3)와 같은 씨어와 데이터 메모리(81) 사이, 및 상기 씨어와 다음과 같은 파이프라인 회로의 구성요소들, 즉, 통신 셸(84)를 통한 상기 하드와이어드 파이프라인(74_1-74_n), 파이프라인 제어기(86), 에외 관리자(88) 및 구성 관리자(90)간의 데이터 전송을 하는 통신 인터페이스(82)가 포함된다. 상기 파이프라인 회로(80)에는 산업-표준 버스 인터페이스(90) 및 상기 인터페이스(82)와 상기 인터페이스(91)를 연결하는 통신 버스(92)가 포함되기도 한다. 선택적으로, 인터페이스(91)의 기능은 통신 인터페이스(82) 내부에 포함되어 있고 버스(93)는 생략된다. 하드와이어드 파이프라인(74_1-74_n), 제어기(86), 예외 관리자(88), 구성 관리자(90) 및 버스 인터페이스(91)의 구조 및 동작은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/683,929호에 설명되어 있다.

통신 인터페이스(82)는 메시지 처리기(64)(도 3)에 의해 확인된 포맷으로 데이터를 송수신해서(존재한다면 버스 인터페이스(91)를 통해서), 일반적으로 피어-벡터 머신(40)(도 3)의 설계 및 수정을 용이하게 한다. 예를 들어, 만일 데이터 포맷이 Rapid I/O 포맷과 같은 산업 표준인 경우, 설계자는 호스트 프로세서(42)와 파이프라인 유닛(78) 사이의 맞춤형 접속을 설계할 필요는 없다. 또한, 논-버스 인터페이스를 통해서가 아니라 파이프라인 버스(60)를 통해서 호스트 프로세서(42)(도 3)와 같은 다른 피어와 파이프라인 유닛(78)이 통신하도록 하게 함으로써, 설계자는 단지 이들을 파이프라인 유닛이 추가되거나 제거되는 시간마다 스크래치로부터 논-버스 인터페이스를 재설계하는 대신 파이프라인 버스와 이들(또는 이들을 포함하고 있는 회로 카드)을 접속하거나 접속해제하는 것만으로 파이프라인 유닛의 수를 변경할 수 있다.

파이프라인 회로(80)는 FPGA와 같은 PLIC이고, 통신 인터페이스(82)에는, 후술하는 바와 같이, 상기 파이프라인 회로로 하여금 상기 호스트 프로세서(42)(도 3)에서부터 상기 펌웨어 메모리(52)로 펌웨어를 로드하게 하는 프로그래밍 포트(94)가 포함되어 있다. 예를 들어, 만일 펌웨어 메모리(52)가 EEPROM 이라면, 프로그래밍 사이클 동안 상기 통신 인터페이스(82)는 상기 펌웨어가 필요로 하는 프로그래밍 신호를 생성하고, 포트(94)가 이를 운반한다. 그러한 프로그래밍 신호를 생성하기 위한 회로는 통상적이어서 본 명세서에서는 설명하지 않는다.

상기 통신 인터페이스(82)의 구조 및 동작은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

도 4를 계속 참고하면, 상기 파이프라인 회로(80)에는 테스트 포트(96), 여기서 상기 파이프라인 회로는 PLIC 임, 및 하드-구성 포트(98)도 포함된다. 상기 테스트 버스(63)와 결합된 상기 테스트 포트(96)는 호스트 프로세서(42)(도 3)로 하여금 파이프라인 회로(80)가 후술하는 바와 같이 피어-벡터 머신(40)(도 3)의 초기화 동안 수행되는 자가 테스트 결과를 모니터하게 한다. 제조자는 보통 파이프라인 회로(80)가 있는 테스트 포트(96)이 포함되어 있고, 일반적으로 상기 테스트 포트를 JTAG 와 같은 산업-표준 테스트 프로토콜과 호환되는 인터페이스(도시하지 않음)로 제공한다. 상기 하드-구성 포트(98)는 후술하는 바와 같이 파이프라인 회로(80)로 하여금 메모리(52)로부터 펌웨어를 다운로드 하여 자기 자신을 구성하도록 한다. 테스트 포트(96)과 유사하게, 제조자는 보통 파이프라인 회로(80)가 있는 구성 포트(98)를 포함시키고, 일반적으로 산업-표준 메모리 인터페이스가 있는 구성 포트 및 상기 메모리(52)의 미리 결정된 어드레스 범위로부터 상기 펌웨어를 직렬로 다운로드하는 상태 머신(도시하지 않음)을 제공한다.

위에서 설명하고 후술하는 바와 같이, 파이프라인 회로(80)가 PLIC 이면, 펌웨어 메모리(52)는 파이프라인 회로의 하나 또는 그 이상의 하드 구성을 나타내는 펌웨어를 저장한다. 상기 펌웨어 메모리에는 테스트 포트(108) 및 프로그래밍 및 구성 포트(106,108)가 포함되어 있다. 상기 테스트 포트(104)는 상기 테스트 버스(63)과 결합되어 있으며, 호스트 프로세서(42)(도 3)으로 하여금, 후술하는 바와 같이 피어-벡터 머신(40)(도 3)의 초기화 동안 펌웨어 메모리(52)가 수행하는 자가 테스트의 결과를 모니터하도록 한다. 아래에 설명하는 바와 같이, 상기 테스트 포트(104)는 호스트 프로세서(42)가 펌웨어를 메모리(52)로 로드하게 한다. 제조자는 일반적으로 테스트 포트(104)에 메모리(52)를 포함시키며, 상기 테스트 포트에 JTAG 와 같은 산업-표준 테스트 프로토콜과 호환되는 인터페이스(도시하지 않음)를 제공한다. 프로그래밍 포트(106)은 프로그래밍 버스(110)를 통해 통신 인터페이스(82)의 프로그래밍 포트(94)와 결합되어 있으며, 후술하는 바와 같이, 파이프라인 회로(80)가 펌웨어를 상기 메모리(52)로 로드하게 한다. 구성 버스(112)를 통해 파이프라인 회로(80)의 하드-구성 포트(98)와 결합되어 있는 하드-구성 포트(108)는, 후술하는 바와 같이, 파이프라인 회로가 메모리(52)로부터 펌웨어를 다운로드하게 한다. 일반적으로, 펌웨어 메모리(52)는 EEPROM 과 같은 비휘발성 메모리로서 전력이 없어도 데이터가 유지된다. 따라서, 펌웨어 메모리(52)는 파이프라인 유닛(78)의 전원이 꺼진 후에도 상기 펌웨어를 계속 저장한다.

도 4를 계속 참고하면, 비록 펌웨어 메모리(52)와 데이터 메모리(81)를 파이프라인 회로(80) 외부에 있는 것으로 설명하였으나, 상기 두 메모리 중 어느 하나 또는 둘 모두는 파이프라인 회로에 통합되기도 한다. 메모리(52)가 파이프라인 회로 내부에 배치되는 경우, 설계자는 프로그래밍 및 구성 버스(110,112)의 구조를 그에 따라 수정할 필요가 있다. 또한, 비록 파이프라인 유닛(78)을 구성 버스(112)와는 분리된 프로그래밍 버스(110)를 가지는 것으로 설명하였으나, 하나의 버스(도시하지 않음)가 상기 프로그래밍 및 구성 버스 모두의 기능을 하기도 한다. 선택적으로, 파이프라인 유닛(78)에는 이 하나의 버스를 위한 다중 인스턴스(multiple instance) 또는 상기 프로그래밍 및 구성(112,110) 중 어느 하나 또는 둘 모두의 다중 인스턴스를 포함하기도 한다.

도 5는 본 발명의 일 실시예에 따른 도 4의 펌웨어 메모리(52)의 논리적 파티셔닝(logical partitioning)의 다이어그램이다.

메모리(52)의 섹션(114)은 파이프라인 회로(80)(도 4)의 초기 구성을 나타내는 펌웨어를 저장하고 있다, 즉, 파이프라인 회로(80)로 다운로드되면, 이 펌웨어는 파이프라인 회로가 상기 초기 구성을 갖게 한다. 상기 초기 구성의 한 구현에서, 파이프라인 회로(80)에는 도 4의 통신 인터페이스(82)(및 필요하다면 산업-표준 버스 인터페이스(91)) 및 상기 파이프라인 회로 및 데이터 메모리(81)의 자가 테스트를 실행하는 자가-테스트 회로(도시하지 않음)를 포함되어 있다. 그래서, 파이프라인 회로(80)는 테스트 버스(63) 또는 통신 인터페이스(82)를 통해 호스트 프로세서(42)(도 3)로 상기 자가 테스트 결과를 제공할 수 있다. 상기 초기 구성은 또한 호스트 프로세서(42)가, 후술하는 바와 같이, 통신 인터페이스(82) 및 프로그래밍 버스(110)를 통해 상기 펌웨어 메모리(52)로 상기 수정된 펌웨어를 로드하게 한다.

메모리(52)의 섹션(116_1-116_i) 각각은 파이프라인 회로(80)의 각각의 동작 구성을 나타내는 펌웨어를 저장하고 있다. 특히, 파이프라인 회로(80)는 가속기(44)(도 3)의 초기화가 끝나는 때에 상기 섹션(116_1-116_i) 중 미리결정된 하나로부터 상기 펌웨어를 다운로드 한다. 후술하는 바와 같이, 파이프라인 회로(80)는 특정한 섹션(116_1-116_n)으로부터 펌웨어를 다운로드하게 프로그램되거나 또는 호스트 프로세서(42)(도 3)은 상기 파이프라인 회로에게 명령하여 특정 섹션으로부터 상기 펌웨어를 다운로드하게 한다. 특히, 1 번째 연산 구조 각각에서, 파이프라인 회로(80)는 도 4에 도시된 구성요소들(예를 들어, 하드와이어드 파이프라인(74_1-74_n))을 포함하고 있다. 그러나, 이들 구성 각각에서, 파이프라인 회로(80)는 일반적으로 서로 다른 동작을 한다. 예를 들어, 통신 인터페이스(82)는 어느 한 구성의 한 프로토콜 및 다른 구성의 다른 프로토콜을 수행한다. 또는, 상기 파이프라인(74_1-74_n)은 어느 한 구성의 데이터상의 연산 세트 하나를 수행하고 다른 구성의 데이터상의 다른 연산 세트를 수행한다.

선택적인 섹션(118)은 상기 메모리(52)의 섹션(116_1-116_i)에 저장된 펌웨어에 의해 각각을 나타내는 동작 구성의 디스크립션 또는 식별을 저장하고 있다. 이 디스크립션/식별은 호스트 프로세서(42)(도 3)가 메모리(52)에 저장된 펌웨어를 식별하게 해준다.

선택적인 섹션(120)은 파이프라인 유닛(78)(도 4)의 프로파일을 저장한다. 이 프로파일은 보통 상기 호스트 프로세서(42)(도 3)가 자기 자신을 적절히 구성하고, 내부간 통신을 위해 피어-엑티 머신(40)(도 3)의 다른 피어 및 파이프라인 유닛으로 상기 파이프라인 유닛(78)의 하드웨어 레이아웃을 충분히 디스크라이브 한다. 예를 들어, 상기 프로파일은 상기 파이프라인 유닛(78)이 수행할 수 있는 데이터 연산 및 통신 프로토콜, 데이터 메모리(81)의 크기, 섹션(116_1-116_i)(섹션(118)이 생략된 경우)내에 저장된 상기 펌웨어에 의해 표현되는 연산 구성, 및 현재 요구되는 연산 구성을 식별한다. 따라서, 피어-엑티 머신(40)의 초기화 동안 상기 프로파일을 판독함으로써, 호스트 프로세서(42)는 상기 메시지 처리기(64)(도 3)를 적절히 구성하여 파이프라인 유닛(78)과 통신할 수 있다. 또한, 호스트 프로세서(42)는 파이프라인 회로(80)가 다운로드해야 하는 펌웨어의 섹션(116_1-116_i)을 선택한다. 또는, 이 펌웨어의 어느 것도 적절하지 않다면, 호스트 프로세서(42)는 수정된 펌웨어를 메모리(52)로 로드한다. 이 기술은 컴퓨터 자신이 디스크 드라이브 등의 새롭게 설치된 주변장치와 통신할 수 있는 "플러그 앤드 플레이" 기술로 알려져 있다.

선택적으로, 섹션(120)은 호스트 프로세서(42)(도 3)가 가속기 구성 레지스트리(70)(도 3) 등에 저장되어 있는 테이블로부터 상기 프로파일을 검색하게 하는 프로파일 식별기 - 보통 "러닝 인덱스(running index)"로 불림 - 를 저장하고 있다, 상기 러닝 인덱스는 보통 상기 호스트 프로세서(42)가 저장된 프로파일과 매치할 수 있는 제품의 모델 번호와 같은 번호이다.

더 다른 대안에서, 파이프라인 유닛(78)(도 4)는 상기 프로파일 식별기를 "하드와이어드" 형태로 저장하여 섹션(120)에 상기 프로파일이 잘못 기록되는 기회를 제거한다. 예를 들어, 파이프라인 유닛(78)은, 호스트 프로세서(80)(도 4)가 테스트 버스(63) 또는 파이프라인 버스(50) 및 파이프라인 회로(80)(도 4)을 통해 판독할 수 있는 하드와이어드 "레지스터" 내에 상기 프로파일 식별기를 저장한다. 이 레지스터는 전자-기계적 스위치, 점퍼, 또는 땜납 접속(도시하지 않음) 등으로 형성된다.

도 5를 계속 참고하면, 펌웨어 메모리(52)의 선택적 섹션(122)는 펌웨어 메모리(52)가 가속기(44)의 초기화 중에 구동되는 자가-테스트 루틴 등의 갖가지 데이터를 저장한다.

도 3 내지 도 5를 참고하면, 피어-엑티 머신(40)의 동작을 - 특히, 호스트 프로세서(42), 파이프라인 회로(80), 및 펌웨어 메모리(52)의 동작 - 본 발명의 일 실시예에 따라 아래에 설명한다.

파어-백터 머신(40)을 커맨, 호스트 프로세서(42)는 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호에 설명된 바와 같이 스스로 초기화를 하고, 가속기(44)가 스스로 부분 초기화를 한다. 보다 특별하게는, 이 부분 초기화 동안, 파이프라인 회로(80)는 메모리(52)의 섹션(144)에서 상기 초기-구성 펌웨어를 다운로드한다. 앞서 설명한 바와 같이, 상기 초기 구성에서, 파이프라인 회로(80)는 최소의 통신 인터페이스(82) 및 테스트 회로(도시하지 않음)를 포함한다. 파이프라인 회로(80)가 상기 초기 구성으로 구성된 후, 상기 테스트 회로는 상기 파이프라인 회로와 데이터 메모리(81)의 자가 테스트를 수행하고, 그 자가 테스트 결과를 테스트 포트(96) 및 테스트 버스(83)를 통해 호스트 프로세서(42)로 제공한다. 상기 펌웨어 메모리(52)도 자가 테스트를 하여 그 결과를, 도 5를 참고하여 위에서 설명한 바와 같이, 테스트 포트(104) 및 테스트 버스(63)을 통해 호스트 프로세서(42)로 제공한다.

다음으로, 호스트 프로세서(42)는 만일 가속기(44)의 부분 초기화 동안 에러가 발생하는지를 결정한다. 예를 들어, 호스트 프로세서(42)는 상기 테스트 버스(63)로부터 상기 자가 테스트 결과를 분석하고 파이프라인 회로(80), 데이터 메모리(81) 및 펌웨어 메모리가 적절한 기능을 하는지를 결정한다.

만일, 에러가 발생했다면, 호스트 프로세서(42)는 이것을 미리 결정된 방법으로 처리한다. 예를 들어, 호스트 프로세서(42)는 파이프라인 회로(80)로부터 자가 테스트 결과를 수신하지 않고, 테스트 버스(63)를 통해 상기 초기 구성 펌웨어가 펌웨어 메모리(52)의 섹션(144)에 저장되어 있는지를 체크한다. 만일, 초기-구성 펌웨어가 저장되어 있지 않다면, 호스트 프로세서(42)는 파이프라인 버스(50) 또는 테스트 버스(63)를 통해 상기 초기-구성 펌웨어를 상기 섹션(144)로 로드하여 파이프라인 회로(80)가 이 펌웨어를 다운로드하게 한 다음 상기 자가 테스트 결과를 분석한다. 호스트 프로세서의 이러한 에러 처리는 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,053호에 더 설명되어 있다.

만일 어떠한 에러도 발생하지 않았다면, 호스트 프로세서(42)는 파이프라인 유닛(78)으로부터 파이프라인 식별기를 판독하고, 상기 가속기 구성 레지스트리(70)로부터 파이프라인 유닛의 대응하는 프로파일을 계속 얻는다. 펌웨어 메모리(52)의 섹션(120)으로부터가 아니라 상기 레지스트리(70)로부터 이 프로파일을 얻는 것이 적절하다고 하는데, 그 이유는 파이프라인 회로(80)가 ASIC 라면 파이프라인 유닛(78)이 펌웨어 메모리와 같은 비휘발성 메모리를 포함하지 않기 때문이다. 만일 프로파일 식별기가 프로파일 회로(80)가 ASIC 임을 나타낸다면, 호스트 프로세서(42)는 어떠한 펌웨어도 파이프라인 회로로 다운로드될 필요가 없다는 결정을 한다. 선택적으로, 호스트 프로세서(42)(도 3)은 펌웨어 메모리(52)의 섹션(120)으로부터 상기 프로파일을 얻어도 좋다. 이 대안에서, 비록 파이프라인 유닛이 상기 프로파일이 섹션(120)에서 부주의로 인해 삭제된 경우 프로파일 식별기를 저장하긴 하지만, 파이프라인 회로(80)는 프로파일 식별기를 저장할 필요가 없다.

다음으로, 파이프라인 유닛(78) 모두(도 4에는 하나만 도시됨)로부터 상기 프로파일 식별기를 판독한 후, 호스트 프로세서(42)는 가속기(44)의 파이프라인 유닛(78) 모두의 맵(map)을 생성하고 이 맵을 처리기 메모리(88) 등에 저장한다.

그리고 나서, 각각의 파이프라인 유닛(78)을 위해, 호스트 프로세서(42)는 상기 프로파일로부터 파이프라인 회로(80)의 원하는 동작 구성의 일치(identity)를 추출한다. 가속기(44)의 초기화 중에 상기 원하는 동작 구성을 추출하는 것은 설계자로 하여금 상기 초기화에 앞서 프로파일을 업데이트 하는 것만으로 파이프라인 회로(80)의 동작을 수정할 수 있게 한다.

다음으로, 호스트 프로세서(42)는 상기 원하는 동작 구성을 나타내는 펌웨어가 상기 펌웨어 메모리(52)에 저장되어 있는지를 결정한다. 예를 들어, 호스트 프로세서(42)는 프로그래밍 버스(110) 및 통신 인터페이스(82)를 통해 - 파이프라인 회로(80)가 초기 구성에 있고, 통신 인터페이스가 존재하기 때문에 - 상기 메모리 섹션(118)로부터 구성 디스크립션을 판독하고, 상기 원하는 펌웨어가 섹션(116_1-116_j) 중 어느 하나에 저장되어 있는지를 결정한다. 선택적으로, 호스트 프로세서(42)는 테스트 버스(63) 및 테스트 포트(104)를 통해 메모리(52)로부터 상기 구성 디스크립션을 직접 판독한다.

만일, 상기 원하는 동작 구성을 나타내는 펌웨어가 펌웨어 메모리(52)내에 저장되어 있지 않다면, 호스트 프로세서(42)는 이 펌웨어를, 통신 인터페이스(82), 프로그래밍 포트(94,106) 및 프로그래밍 버스(110)를 통해, 상기 구성 레지스트리(70)에서부터 상기 펌웨어 메모리의 섹션(116_1-116_j) 중 하나로 로드한다. 만일, 상기 펌웨어가 레지스트리(70)에 없으면, 호스트 프로세서(42)는 외부 라이브러리(도시하지 않음)로부터 상기 펌웨어를 검색하거나 또는 에러 표시기를 생성하여 시스템 연산(도시하지 않음)이 상기 펌웨어를 상기 레지스트리(70)로 로드할 수 있게 한다.

다음으로, 호스트 프로세서(42)는 파이프라인 회로(80)에 명령을 하여 상기 포트(108), 구성 버스(112) 및 포트(98)를 통해 상기 메모리(52)의 대응하는 섹션(116_1-116_t)으로부터 원하는 펌웨어를 다운로드 한다.

파이프라인 회로(80)가 상기 원하는 펌웨어를 다운로드한 후, 이를 상기 원하는 동작 구성내에 있게 하고 데이터 처리를 시작할 준비를 한다. 그러나, 파이프라인 회로(80)가 상기 원하는 동작 구성에 있은 후에라도, 호스트 프로세서(42)는 통신 인터페이스(82)를 통해 또는 테스트 버스(63)를 통해 상기 메모리(52)의 상기 섹션(116_1-116_t)으로 새로운 펌웨어를 로드한다. 예를 들어, 새로운 펌웨어를 로드하기 위해서, 호스트 프로세서(42)는 상기 파이프라인 회로(80)가 상기 메모리(52)의 섹션(114)으로부터 상기 펌웨어를 재로드하도록 하여 파이프라인 회로가 상기 초기 구성으로 다시 가도록 한다. 그러한, 호스트 프로세서(42)는 파이프라인 버스(50)를 통해 또는 통신 인터페이스(82)를 통해서 상기 섹션(116_1-116_t) 중 하나로 상기 새로운 펌웨어를 로드한다. 다음으로, 호스트 프로세서(42)는 상기 파이프라인 회로(80)가 새로운 펌웨어를 다운로드하게 하여 파이프라인 회로가 새로운 동작 구성에 있도록 한다. 상기 초기 구성이 아래의 두 가지 장점을 제공하는 경우에만, 파이프라인 회로(80)가 새로운 펌웨어를 메모리(52)로 프로드하게 한다. 첫번째로, 파이프라인 회로(80)가 동작 구성에 있는 경우, 파이프라인 회로(80)가 메모리(52)내에 저장된 펌웨어를 부주의로 바꾸는 것을 막아준다. 두번째로, 상기 동작 구성이, 펌웨어를 메모리(52)로 기록하는 것을 요구하는 회로를 위해 사용되는 파이프라인 회로(80)의 리소스를 사용하도록 한다.

도 6은 본 발명의 일 실시예에 따른 도 3의 파이프라인 가속기(44)의 파이프라인 유닛(124)의 블럭 다이어그램이다.

파이프라인 유닛(124)은, 다수의 파이프라인 회로(80) - 여기에서는 두 개의 파이프라인 회로(80a, 80b) - 및 다수의 펌웨어 메모리 - 여기에서는 두 개의 메모리(52a,52b), 각각의 파이프라인 회로를 위한 메모리 - 가 포함되어 있다는 것을 제외하고는 도 4의 파이프라인 유닛(78)과 유사하다. 상기 파이프라인 회로(80a,80b)와 펌웨어 메모리(52a,52b)의 결합은 프로그램가능한 회로 유닛을 형성한다. 한 구현에서, 상기 메모리(52a,52b) 각각은, 펌웨어 메모리(52b)가 상기 파이프라인 유닛(124)의 프로파일을 저장하고 그렇지 않으면 메모리(52a)의 섹션(120)으로 동작하는, 섹션(120)을 생략할 수 있다는 것을 제외하고는 도 5에 도시된 바와 같이 파티션된다. 선택적으로, 파이프라인 회로(80a,80b)는 상기 메모리(52a,52b)와 유사하게 동작하는 각각의 섹션을 포함하는 하나의 펌웨어 메모리를 할당한다. 파이프라인 회로(80)의 수를 늘이는 것은 하드와이어드 파이프라인(74_1-74_n)의 개수(n)를 증가시켜 파이프라인 유닛(78)과 비교할 때 파이프라인 유닛(124)의 기능을 증가시킨다. 또한, 대응하는 펌웨어 메모리(52)가 생략 가능한 경우에는, 파이프라인 회로(80a,80b)의 하나 또는 둘 모두가 ASIC 이어도 좋다.

파이프라인 유닛(124)의 구조 및 동작에 대한 보다 상세한 설명은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/683,929호에 설명되어 있다.

파이프라인 회로(80a)에는 테스트 포트(96a) 및 하드-구성 포트(98a)가 포함되어 있고, 이들 각각은 도 4의 테스트 포트(96) 및 하드-구성 포트(98)와 각각 유사하다. 그리고, 도 4의 파이프라인 회로(80)와 같이, 상기 파이프라인 회로(80a)에는 프로그래밍 포트(94)를 가지는 통신 인터페이스(82)가 포함되어 있다.

파이프라인 회로(80b)에는 테스트 포트(96b) 및 하드-구성 포트(98b)가 포함되어 있는데, 이들 각각은 도 4의 테스트 포트(96) 및 하드-구성 포트(98)와 각각 유사하다. 그리고, 호스트 프로세서(42)(도 3)가, 후술하는 바와 같이, 파이프라인 회로(80a)의 통신 인터페이스(82)를 통해 펌웨어 메모리(52)를 프로그램할 수 있기 때문에, 상기 파이프라인 회로(80b)에는 프로그래밍 포트(94)가 포함되어 있지 않다.

펌웨어 메모리(52a)에는 테스트, 프로그래밍 및 하드-구성 포트(104a,106a,108a)가 포함되어 있는데, 이들 각각은 도 4의 테스트, 프로그래밍 및 하드-구성 포트(104,106,108)과 각각 유사하다. 테스트 포트(104a)는 테스트 버스(63)와 결합되고, 프로그래밍 포트(106a)는 프로그래밍 버스(110)를 통해 통신 인터페이스(82)의 프로그래밍 포트(94a)와 결합되어 있으며, 하드-구성 포트(108a)는 구성 버스(112a)를 통해 파이프라인 회로(80a)의 하드-구성 포트(98a)와 결합된다.

유사하게, 펌웨어 메모리(52b)에는 테스트, 프로그래밍, 및 하드-구성 포트(104b,106b,108b)가 포함되는데, 이들 각각은 도 4의 테스트, 프로그래밍, 및 하드-구성 포트(104,106,108)과 각각 유사하다. 테스트 포트(104b)는 테스트 버스(63)

와 결합되어 있고, 프로그래밍 포트(106b)는 프로그래밍 버스(110)를 통해 통신 인터페이스(82)의 프로그래밍 포트(94a)와 결합되어 있고, 하드-구성 포트(108b)는 구성 버스(112b)를 통해 파이프라인 회로(80b)의 하드-구성 포트(98b)와 결합되어 있다.

도 3을 참고하면, 퍼어-벡터 머신(40)의 동작 - 특히 호스트 프로세서(42), 파이프라인 회로(80a,80b) 및 펌웨어 메모리(52a) - 을 본 발명의 일 실시예에 따라 아래에 설명한다.

퍼어-벡터 머신(40)이 켜지면, 호스트 프로세서(42)는, 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,053호에 설명된 바와 같이, 자기 자신을 초기화 하고, 가속기(44)가 자기 자신을 부분적으로 초기화 한다. 보다 특별하게는, 이 부분 초기화 동안, 파이프라인 회로(80a,80b)는 펌웨어 메모리(52a,52b)의 섹션(144a,114b)에서 상기 초기-구성 펌웨어를 다운로드한다. 각각의 초기 구성에서, 파이프라인 회로(80a)는 최소의 통신 인터페이스(82) 및 테스트 회로(도시하지 않음)를 포함하고, 파이프라인 회로(80b)는 최소 테스트 회로(도시하지 않음)를 포함한다. 파이프라인 회로(80a,80b)가 그들 각각의 초기 구성으로 구성된 후, 각각의 파이프라인 회로 내부의 상기 테스트 회로는 상기 파이프라인 회로의 각각의 자기 테스트를 수행하고 - 파이프라인 회로(80a,80b)의 하나 또는 둘 모두의 테스트 회로는 데이터 메모리(81)도 테스트함 - 그 자기 테스트 결과를 테스트 포트(96a,96b) 각각을 통해 및 테스트 버스(63)를 통해 호스트 프로세서(42)로 제공한다. 상기 펌웨어 메모리(52a,52b)도 자기 테스트를 하여 그 결과를, 도 5를 참고하여 위에서 설명한 바와 같이, 테스트 포트(104a,104b) 각각을 통해서 및 테스트 버스(63)을 통해서 호스트 프로세서(42)로 제공한다.

다음으로, 호스트 프로세서(42)는 만일 가속기(44)의 부분 초기화 동안 에러가 발생하는지를 결정한다. 예를 들어, 호스트 프로세서(42)는 상기 테스트 버스(63)로부터 상기 자기 테스트 결과를 분석하고 파이프라인 회로(80a,80b), 데이터 메모리(81) 및 펌웨어 메모리(52a,52b)가 적절한 기능을 하는지를 결정한다.

만일, 에러가 발생했다면, 호스트 프로세서(42)는 이것을 미리 결정된 방법으로 처리한다. 예를 들어, 호스트 프로세서(42)는 파이프라인 회로(80a)로부터 자기 테스트 결과를 수신하기 않고, 테스트 버스(63)를 통해 상기 초기 구성 펌웨어가 펌웨어 메모리(52)의 섹션(114a)에 저장되어 있는지를 체크한다. 만일, 초기-구성 펌웨어가 저장되어 있지 않다면, 호스트 프로세서(42)는 파이프라인 버스(60) 또는 테스트 버스(63)를 통해 상기 초기-구성 펌웨어를 상기 섹션(114a)로 로드하여 파이프라인 회로(80)가 이 펌웨어를 다운로드하게 한 다음 상기 자기 테스트 결과를 분석한다. 이 실시예는 또한 파이프라인 회로(50b) 및 펌웨어 메모리(52b)에도 적용된다. 호스트 프로세서의 이러한 에러 처리는 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,053호에 더 설명되어 있다.

만일 어떠한 에러도 발생하지 않았다면, 호스트 프로세서(42)는 파이프라인 유닛(124)으로부터 프로파일 식별기를 판독하고, 상기 가속기 구성 레지스트리(70)로부터 또는 도 4를 참고하여 앞서 설명한 바와 같이 펌웨어 메모리(52)의 섹션(12)으로부터 파이프라인 유닛의 대응하는 프로파일을 계속 얻는다.

다음으로, 파이프라인 유닛(124) 모두(도 4에는 하나만 도시됨)로부터 상기 프로파일 식별기를 판독한 후, 호스트 프로세서(42)는 가속기(44)의 파이프라인 유닛 모두의 맵(map)을 생성하고 이 맵을 처리기 메모리(68) 등에 저장한다.

그리고 나서, 호스트 프로세서(42)는 상기 프로파일로부터 파이프라인 회로(80a,80b)의 원하는 동작 구성의 일치(identity)를 추출한다. 가속기(44)의 초기화 중에 상기 원하는 동작 구성을 추출하는 것은 설계자로 하여금 상기 초기화에 앞서 프로파일을 업데이트 하는 것만으로 파이프라인 회로(80a 및/또는 80b)의 동작을 수정할 수 있게 한다.

다음으로, 호스트 프로세서(42)는 상기 원하는 동작 구성을 나타내는 펌웨어가 상기 펌웨어 메모리(52a,52b)에 저장되어 있는지를 결정한다. 예를 들어, 호스트 프로세서(42)는 프로그래밍 버스(110) 및 통신 인터페이스(82)를 통해 - 파이프라인 회로(80a)가 초기 구성에 있고, 통신 인터페이스가 존재하기 때문에 - 상기 메모리 섹션(118a)로부터 구성 디스크립션을 판독하고, 상기 원하는 펌웨어가 섹션(116_1-116_j) 중 어느 하나에 저장되어 있는지를 결정한다. 선택적으로, 호스트 프로세서(42)는 테스트 버스(63) 및 테스트 포트(104a)를 통해 메모리(52a)로부터 상기 구성 디스크립션을 직접 판독한다. 이 예는 파이프라인 회로(50b) 및 펌웨어 메모리(52b)에 적용되기도 한다.

만일, 상기 원하는 동작 구성을 나타내는 펌웨어가 펌웨어 메모리(52a 및/또는 52b)내에 저장되어 있지 않다면, 호스트 프로세서(42)는 이 펌웨어를, 통신 인터페이스(82), 프로그래밍 포트(94,106) 및 프로그래밍 버스(110)를 통해, 상기 구성 레지스트리(70)에서부터 상기 펌웨어 메모리의 섹션(116_1-116_j) 중 하나로 로드한다. 예를 들어, 만일 파이프라인 회로

(80a)의 원하는 동작 구성을 나타내는 펌웨어가 메모리(52b)내에 저장되어 있지 않다면, 호스트 프로세서(42)는 인터페이스(82), 프로그래밍 포트(94,106b) 및 프로그래밍 버스(110)를 통해 이 펌웨어를 레즈스트리(70)로부터 섹션($116b_1$-$116b_j$) 의 하나로 로드한다. 반면, 상기 펌웨어가 레지스트리(70)에 없으면, 호스트 프로세서(42)는 외부 라이브러리(도시하지 않음)로부터 상기 펌웨어를 검색하거나 또는 예외 표시기를 생성하여 시스템 연산(도시하지 않음)이 상기 펌웨어를 상기 레지스트리(70)로 로드할 수 있게 한다.

다음으로, 호스트 프로세서(42)는 파이프라인 회로(80a)에 명령을 하여 상기 포트(108a), 구성 버스(112a) 및 포트(98a)를 통해 상기 메모리(52a)의 대응하는 섹션($116a_1$-$116a_j$)으로부터 원하는 펌웨어를 다운로드 하고, 파이프라인 회로(80b)에 명령을 하여 포트(108b), 구성 버스(112b) 및 포트(98b)를 통해 메모리(52b)의 대응하는 섹션($116b_1$-$116b_j$)으로부터 상기 원하는 펌웨어를 다운로드 하게 한다.

파이프라인 회로(80a,80b)가 상기 원하는 펌웨어를 다운로드한 후, 이를 상기 원하는 동작 구성내에 있게 하고 데이터 처리를 시작할 준비를 한다. 그러나, 파이프라인 회로(80a,80b)가 상기 원하는 동작 구성에 있은 후에라도, 호스트 프로세서(42)는 통신 인터페이스(82)를 통해 또는 테스트 버스(63)를 통해, 도 4를 참고하여 상기 설명한 바와 유사한 방식으로, 상기 메모리(52a,52b)의 상기 섹션(116_1-116_j)으로 새로운 펌웨어를 로드한다.

앞의 설명으로부터 당업자는 본 발명을 실시하거나 활용할 수 있다. 본 발명의 실시예들을 당업자는 다양하게 변형시킬 수 있다는 것은 분명하고, 본 발명의 요지와 범위를 벗어나지 않고 다른 실시예 및 응용 분야에 적용할 수 있다. 따라서, 본 발명의 실시예들은 본 발명을 제한하기 위한 것이 아니고 여기에서 공개한 원리와 특징에 넓게 해석되야 할 것이다.

(57) 청구의 범위

청구항 1.

외부 소스로부터, 구성을 나타내는 펌웨어를 수신하고;

상기 펌웨어를 메모리에 저장하고; 그리고

상기 펌웨어를 상기 메모리로부터 다운로드 하도록 동작 가능한 것을 특징으로 하는 프로그램가능한 회로.

청구항 2.

청구항 1에 있어서,

상기 메모리로부터 상기 펌웨어를 다운로드한 후 상기 구성내에서 연산을 하도록 동작 가능한 것을 특징으로 하는 프로그램가능한 회로.

청구항 3.

청구항 1에 있어서,

상기 메모리는 비휘발성 메모리를 포함하는 것을 특징으로 하는 프로그램가능한 회로.

청구항 4.

청구항 1에 있어서,

상기 메모리는 외부 메모리를 포함하는 것을 특징으로 하는 프로그램가능한 회로.

청구항 5.

제1 구성을 나타내는 제1 펌웨어를 메모리로부터 다운로드 하고;

상기 제1 구성을 연산하고;

제2 구성을 나타내는 제2 펌웨어를 상기 메모리로부터 다운로드 하고; 그리고

상기 제2 메모리내에서 연산하도록 동작 가능한 것을 특징으로 하는 프로그램가능한 회로.

청구항 6.

청구항 5에 있어서,

상기 프로그램가능한 회로는:

상기 제1 구성이 연산되는 동안 외부 소스로부터 상기 제2 펌웨어를 수신하고; 그리고

상기 제1 구성이 연산되는 동안 상기 메모리로부터 상기 제2 펌웨어를 저장하도록 동작 가능한 것을 특징으로 하는 프로그램가능한 회로.

청구항 7.

메모리; 및

상기 메모리에 결합된 프로그램가능한 회로를 구비하고,

상기 프로그램가능한 회로는,

외부 소스로부터, 상기 프로그램가능한 회로의 구성을 나타내는 펌웨어를 수신하고;

상기 펌웨어를 상기 메모리에 저장하고; 그리고

상기 펌웨어를 상기 메모리로부터 다운로드 하도록 동작 가능한 것을 특징으로 하는 프로그램가능한 회로.

청구항 8.

청구항 7에 있어서,

상기 메모리는 전기적으로 디스에이블 할 수 있고 프로그램가능한 판독-전용 메모리(ROM)을 포함하는 것을 특징으로 하는 프로그램가능한 회로.

청구항 9.

청구항 7에 있어서,

상기 프로그램가능한 회로는 필드-프로그램가능한 게이트 어레이를 포함하는 것을 특징으로 하는 프로그램가능한 회로.

청구항 10.

각각 제1 및 제2 구성을 나타내는 제1 및 제2 펌웨어 데이터를 저장하도록 동작 가능한 메모리; 및

상기 메모리에 결합된 프로그램가능한 회로를 구비하고,

상기 프로그램가능한 회로는,

상기 메모리로부터 상기 제1 펌웨어를 다운로드하고,

상기 제1 구성내에서 연산하고,

상기 메모리로부터 상기 제2 펌웨어를 다운로드하고, 그리고

상기 제2 구성내에서 연산하도록 동작 가능한 것을 특징으로 하는 프로그램가능한 회로.

청구항 11.

청구항 10에 있어서,

상기 프로그램가능한 회로는:

상기 제1 구성내에서 연산하는 동안 외부 소스로부터 상기 제2 펌웨어를 수신하고; 그리고

상기 제1 구성내에서 연산하는 동안 상기 메모리로부터 상기 제2 펌웨어를 저장하도록 동작 가능한 것을 특징으로 하는 프로그램가능한 회로.

청구항 12.

청구항 10에 있어서,

상기 프로그램가능한 회로는 상기 제1 구성내에서 연산하는 동안 상기 제2 펌웨어를 프로하도록 동작 가능한 것을 특징으로 하는 프로그램가능한 회로.

청구항 13.

각각 제1, 제2, 제3 및 제4 구성을 나타내는 제1, 제2, 제3 및 제4 펌웨어를 저장하도록 동작 가능한 메모리;

상기 메모리에 결합되어 있는 제1 프로그램가능한 회로; 및

상기 메모리와 상기 제1 프로그램가능한 회로에 결합되어 있는 제2 프로그램가능한 회로를 구비하고,

상기 제1 프로그램가능한 회로는,

상기 메모리로부터 상기 제1 펌웨어를 다운로드하고,

상기 제2 구성내에서 연산을 하고,

상기 메모리로부터 상기 제2 펌웨어를 다운로드하고, 그리고,

상기 제2 구성내에서 연산을 하도록 동작 가능하며,

상기 제2 프로그램가능한 회로는,

상기 메모리로부터 상기 제3 펌웨어 데이터를 다운로드하고,

상기 제3 구성내에서 연산을 하고,

상기 메모리로부터 상기 제4 펌웨어를 다운로드하고, 그리고,

상기 제4 구성내에서 연산을 하도록 동작 가능한 것을 특징으로 하는 프로그램가능한 회로 유닛.

청구항 14.

청구항 13에 있어서,

상기 제1 프로그램가능한 회로는:

상기 제1 구성내에서 연산하는 동안 외부 소스로부터 상기 제2 및 상기 제4 펌웨어를 수신하고; 그리고

상기 제1 구성내에서 연산하는 동안 상기 메모리 내에 상기 제2 및 상기 제4 펌웨어를 저장하도록 동작 가능한 것을 특징으로 하는 프로그램가능한 회로 유닛.

청구항 15.

청구항 13에 있어서,

상기 제1 및 제2 프로그램가능한 회로는 각각의 필드-프로그램가능한 게이트 어레이를 포함하는 것을 특징으로 하는 프로그램가능한 회로 유닛.

청구항 16.

프로세서; 및

상기 프로세서에 결합된 프로그램가능한 회로 유닛을 구비하고,

상기 프로그램가능한 회로 유닛은,

메모리, 및

상기 메모리와 결합되어 있는 프로그램가능한 회로를 구비하고,

상기 프로그램가능한 회로는,

상기 프로그램가능한 회로의 구성을 나타내는 펌웨어를 상기 프로세서로부터 수신하고,

상기 메모리 내에 상기 펌웨어를 저장하고, 그리고

상기 프로세서에 응답하여 상기 메모리로부터 상기 펌웨어를 다운로드하도록 동작 가능한 것을 특징으로 하는 컴퓨팅 머신.

청구항 17.

청구항 16에 있어서,

상기 프로세서는:

상기 프로그램가능한 회로로 상기 펌웨어를 전송하기 전에 상기 펌웨어가 이미 상기 메모리에 저장되어 있는지를 결정하고, 그리고

상기 펌웨어가 상기 메모리에 이미 저장되어 있지 않은 경우에만 상기 프로그램가능한 회로로 상기 펌웨어를 전송하도록 동작 가능한 것을 특징으로 하는 컴퓨팅 머신.

청구항 18.

청구항 16에 있어서,

상기 프로세서와 결합되어 있고 상기 펌웨어를 저장하도록 및 상기 펌웨어가 상기 프로그램가능한 회로를 위한 원하는 구성을 나타내는지를 나타내도록 동작 가능한 구성 레지스트리를 더 구비하고,

상기 프로세서는, 상기 구성 레지스트리로부터 상기 프로그램가능한 회로까지 상기 펌웨어를 다운로드하도록 동작 가능한 것을 특징으로 하는 컴퓨팅 머신.

청구항 19.

청구항 16에 있어서,

상기 프로그램가능한 회로 유닛은 파이프라인 유닛을 포함하고 있으며,

상기 프로그램가능한 회로에는 데이터상에서 연산되도록 동작 가능한 하드와이어드 파이프라인이 포함되어 있는 것을 특징으로 하는 컴퓨팅 머신.

청구항 20.

프로세서; 및

상기 프로세서와 결합되어 있는 프로그램가능한 회로 유닛을 구비하고,

상기 프로그램가능한 회로 유닛은,

각각 제1 및 제2 구성을 나타내는 제1 및 제2 펌웨어를 저장하도록 동작 가능한 메모리, 및

상기 메모리로부터 상기 제1 펌웨어를 다운로드하고,

상기 제1 구성내에서 연산하고,

상기 프로세서에 응답하여 상기 메모리로부터 상기 제2 펌웨어를 다운로드하고, 그리고

상기 제2 구성내에서 연산하도록 동작 가능한 프로그램가능한 회로를 구비하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 21.

청구항 20에 있어서,

상기 프로세서는 제1 포트를 포함하고,

상기 프로그램가능한 회로 유닛은 상기 제1 테스트 포트와 결합된 제2 테스트 포트를 포함하며,

상기 프로세서는 상기 제1 및 제2 테스트 포트를 통해 상기 제1 펌웨어를 메모리로 로드하도록 동작 가능한 것을 특징으로 하는 컴퓨팅 머신.

청구항 22.

청구항 20에 있어서,

상기 프로세서는 제1 테스트 포트를 포함하고,

상기 프로그램가능한 회로 유닛은 상기 제1 테스트 포트에 결합된 제2 테스트 포트를 포함하고,

상기 제1 구성에서의 연산 동안, 상기 프로그램가능한 회로는 자가 테스트를 수행하고 상기 제1 및 제2 테스트 포트를 통해 상기 프로세서로 자가 테스트 데이터를 제공하도록 동작 가능한 것을 특징으로 하는 컴퓨팅 머신.

청구항 23.

청구항 20에 있어서,

상기 프로세서는 상기 제2 펌웨어를 상기 프로그램가능한 회로로 전송하도록 동작 가능하며,

상기 제1 구성에서의 연산 동안, 상기 프로그램가능한 회로는 상기 프로세서에 응답하여 상기 제2 펌웨어를 상기 메모리로 로드하도록 동작 가능한 것을 특징으로 하는 컴퓨팅 머신.

청구항 24.

프로세서; 및

상기 프로세서와 결합된 프로그램가능한 회로 유닛을 구비하고,

상기 프로그램가능한 회로 유닛은,

각각 제1, 제2, 제3 및 제4 구성을 나타내는 제1, 제2, 제3 및 제4 펌웨어를 저장하도록 동작 가능한 메모리,

상기 메모리에 결합되고, 상기 메모리로부터 상기 제1 펌웨어를 다운로드하고,

상기 제1 구성내에서 연산하고, 상기 프로세서에 응답하여 상기 메모리로부터 상기 제2 펌웨어를 다운로드하고, 상기 제2 구성내에서 연산하도록 동작 가능한 제1 프로그램가능한 회로; 및

상기 메모리 및 상기 제1 프로그램가능한 회로에 결합되고, 상기 메모리로부터 상기 제3 펌웨어를 다운로드하고, 상기 제3 구성내에서 연산하고, 상기 프로세서에 응답하여 상기 메모리로부터 상기 제4 펌웨어를 다운로드하고, 상기 제4 구성내에서 연산하도록 동작 가능한 제2 프로그램가능한 회로를 구비하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 25.

청구항 24에 있어서,

상기 프로세서는 제1 테스트 포트를 포함하고,

상기 프로그램가능한 회로 유닛은 상기 제1 테스트 포트에 결합된 제2 테스트 포트를 포함하고,

상기 프로세서는 상기 제1 및 제2 테스트 포트를 통해 상기 제1 및 제3 펌웨어를 메모리로 로드하도록 동작 가능한 것을 특징으로 하는 컴퓨팅 머신.

청구항 26.

청구항 24에 있어서,

상기 프로세서는 제1 테스트 포트를 포함하고,

상기 프로그램가능한 회로 유닛은 상기 제1 테스트 포트에 결합된 제2 테스트 포트를 포함하고,

상기 제1 구성에서의 연산 동안, 상기 프로그램가능한 회로는 제1 자가 테스트를 수행하고 상기 제1 및 제2 테스트 포트를 통해 상기 프로세서로 제1 자가 테스트 데이터를 제공하고,

상기 제3 구성에서의 연산 동안, 상기 프로그램가능한 회로는 제2 자가 테스트를 수행하고 상기 제1 및 제2 테스트 포트를 통해 상기 프로세서로 제2 자가 테스트 데이터를 제공하고,

상기 프로세서는 상기 제1 및 제2 프로그램가능한 회로가, 상기 제1 및 제2 자가 테스트 데이터가 상기 제1 및 제2 자가 테스트 각각의 미리결정된 결과를 나타내는 경우에만 상기 메모리로부터 상기 제2 및 제4 펌웨어를 로드하도록 동작 가능한 것을 특징으로 하는 컴퓨팅 머신.

청구항 27.

청구항 24에 있어서,

상기 프로세서는 상기 제2 및 제4 펌웨어를 상기 제1 프로그램가능한 회로로 전송하도록 동작 가능하며,

상기 제1 구성에서의 연산동안, 상기 제1 프로그램가능한 회로는 상기 프로세서에 응답하여 상기 제2 및 제4 펌웨어를 상기 메모리로 로드하도록 동작 가능한 것을 특징으로 하는 컴퓨팅 머신.

청구항 28.

청구항 24에 있어서,

상기 메모리는:

상기 제1 프로그램가능한 회로와 결합되고 상기 제1 및 제2 펌웨어를 저장하도록 동작가능한 제1 메모리 섹션; 및

상기 제1 및 제2 프로그램가능한 회로와 결합되고 상기 제3 및 제4 펌웨어를 저장하도록 동작 가능한 제2 메모리 섹션을 포함하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 29.

청구항 28에 있어서,

상기 제1 및 제2 메모리 섹션은 각각 제1 및 제2 집적회로상에 배치되는 것을 특징으로 하는 컴퓨팅 머신.

청구항 30.

프로그램가능한 회로에, 상기 회로의 구성을 나타내는 펌웨어를 제공하는 단계;

상기 프로그램가능한 회로를 가지고 상기 구성 데이터를 메모리에 저장하는 단계; 및

상기 메모리로부터 상기 구성 데이터를 상기 프로그램가능한 회로로 다운로드하는 단계를 구비하는 것을 특징으로 하는 방법.

청구항 31.

청구항 30에 있어서,

상기 메모리로부터 상기 구성 데이터의 다운로드 후에 상기 구성에서의 연산을 하는 단계를 더 구비하는 것을 특징으로 하는 방법.

청구항 32.

제1 구성을 나타내는 제1 펌웨어를 프로그램가능한 회로로 다운로드하는 단계;

상기 제1 구성에서 상기 프로그램가능한 회로를 동작시키는 단계;

제2 구성을 나타내는 제2 펌웨어를 상기 프로그램가능한 회로로 다운로드하는 단계; 및

상기 제2 펌웨어의 다운로드 후에 상기 제2 구성에서 상기 프로그램가능한 회로를 동작시키는 단계를 구비하는 것을 특징으로 하는 방법.

청구항 33.

청구항 32에 있어서,

상기 제2 펌웨어를 다운로드하는 단계는,

상기 제2 펌웨어를 상기 프로그램가능한 회로로 전송하는 단계;

상기 프로그램가능한 회로가 상기 제1 구성에서 동작하는 동안 상기 프로그램가능한 회로를 가지고 상기 제2 펌웨어를 메모리로 로드하는 단계; 및

상기 메모리로부터 상기 제2 펌웨어를 상기 프로그램가능한 회로로 다운로드하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 34.

상기 제2 펌웨어를 다운로드하는 단계는,

상기 제2 펌웨어가 상기 프로그램가능한 회로와 연결된 메모리 내에 저장되어 있는지를 결정하는 단계;

상기 제2 펌웨어가 상기 메모리 내에 저장되어 있지 않는 경우에만 상기 제2 펌웨어를 상기 프로그램가능한 회로로 전송하는 단계;

상기 프로그램가능한 회로가 상기 제1 구성에서 동작하는 동안 상기 프로그램가능한 회로를 가지고 상기 제2 펌웨어를 상기 메모리로 로드하는 단계; 및

상기 메모리로부터 상기 제2 펌웨어를 상기 프로그램가능한 회로로 다운로드하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 35.

청구항 32에 있어서,

상기 제1 구성에서 상기 프로그램가능한 회로를 동작시키는 단계는 상기 프로그램가능한 회로를 테스트하는 단계를 포함하고,

상기 제2 펌웨어를 다운로드하는 단계는 상기 프로그램가능한 회로가 상기 테스트를 통과하는 경우에만 상기 제2 펌웨어를 다운로드하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 36.

제1 및 제2 펌웨어를 제1 및 제2 프로그램가능한 회로로 각각 다운로드하는 단계;

상기 제1 및 제2 구성에서 상기 제1 및 제2 프로그램가능한 회로를 각각 동작시키는 단계;

상기 프로그램가능한 회로를 통해, 상기 제3 및 제4 펌웨어를 상기 제1 및 제2 프로그램가능한 회로로 각각 다운로드하는 단계; 및

상기 제3 구성 및 제4 구성에서 상기 제1 및 제2 프로그램가능한 회로를 각각 동작시키는 단계를 구비하는 것을 특징으로 하는 방법.

청구항 37.

청구항 36에 있어서,

상기 제1 및 제2 펌웨어를 다운로드하는 단계는, 테스트 포트를 통해 상기 제1 및 제2 펌웨어를 상기 제1 및 제2 프로그램가능한 회로로 각각 다운로드하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 38.

청구항 36에 있어서,

상기 제1 및 제2 구성에서 상기 제1 및 제2 프로그램가능한 회로를 동작시키는 단계는 상기 제1 및 제2 프로그램가능한 회로를 테스트하는 단계를 포함하고,

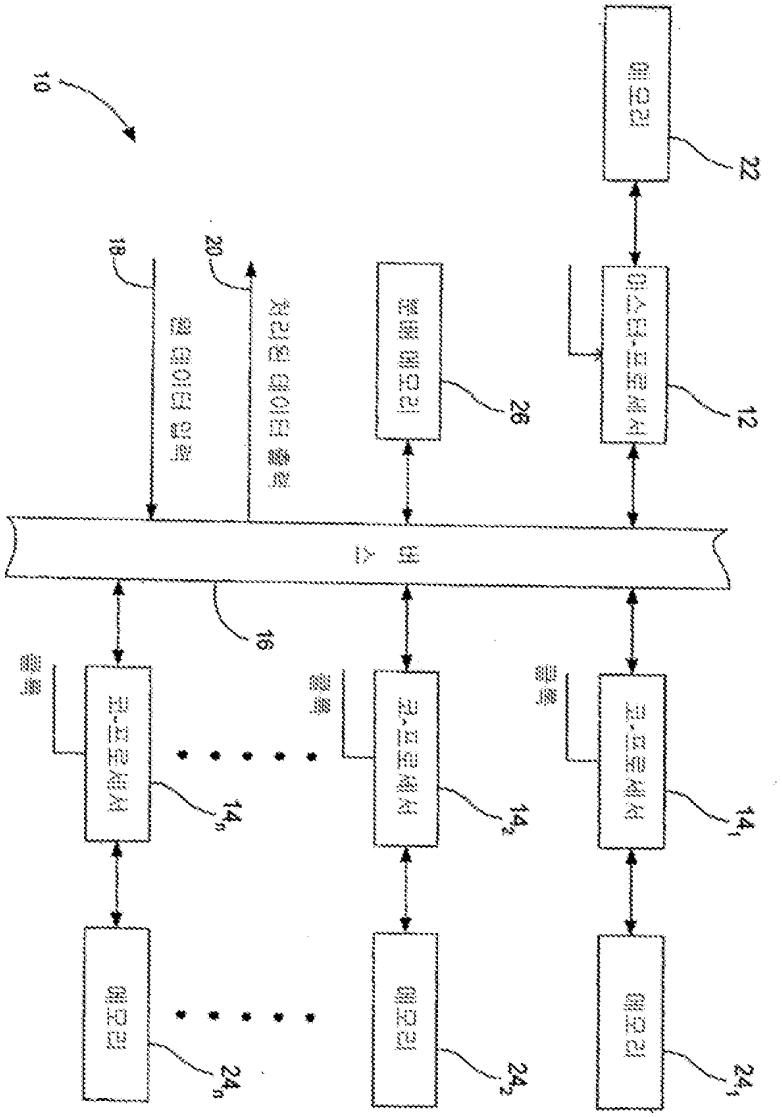
상기 제3 및 제4 펌웨어를 상기 제1 및 제2 프로그램가능한 회로로 로드하는 단계는,

상기 테스트가 상기 제1 프로그램가능한 회로가 원하는 기능을 하는 경우에만 상기 제3 펌웨어를 로드하는 단계, 및

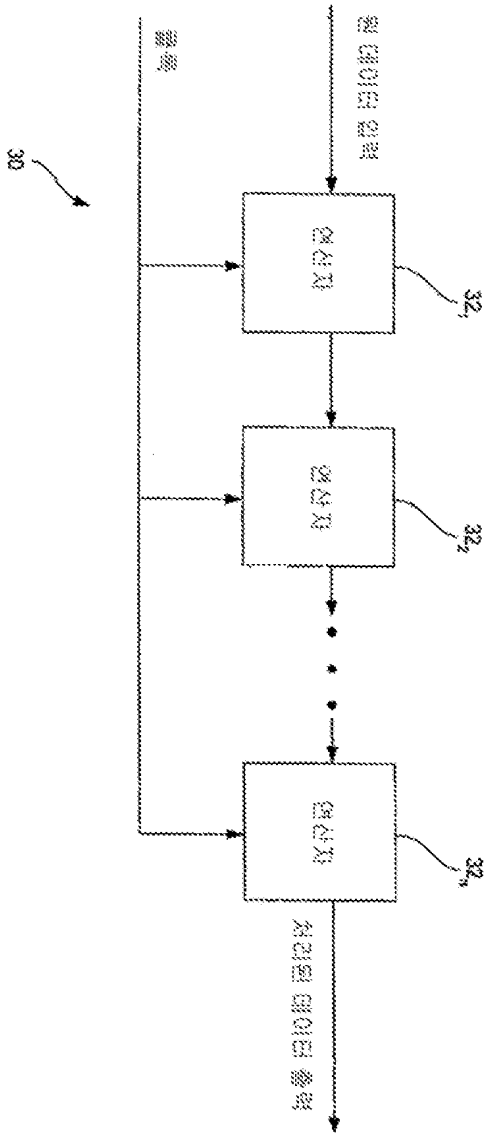
상기 테스트가 상기 제2 프로그램가능한 회로가 원하는 기능을 하는 경우에만 상기 제4 펌웨어를 로드하는 단계를 포함하는 것을 특징으로 하는 방법.

도면

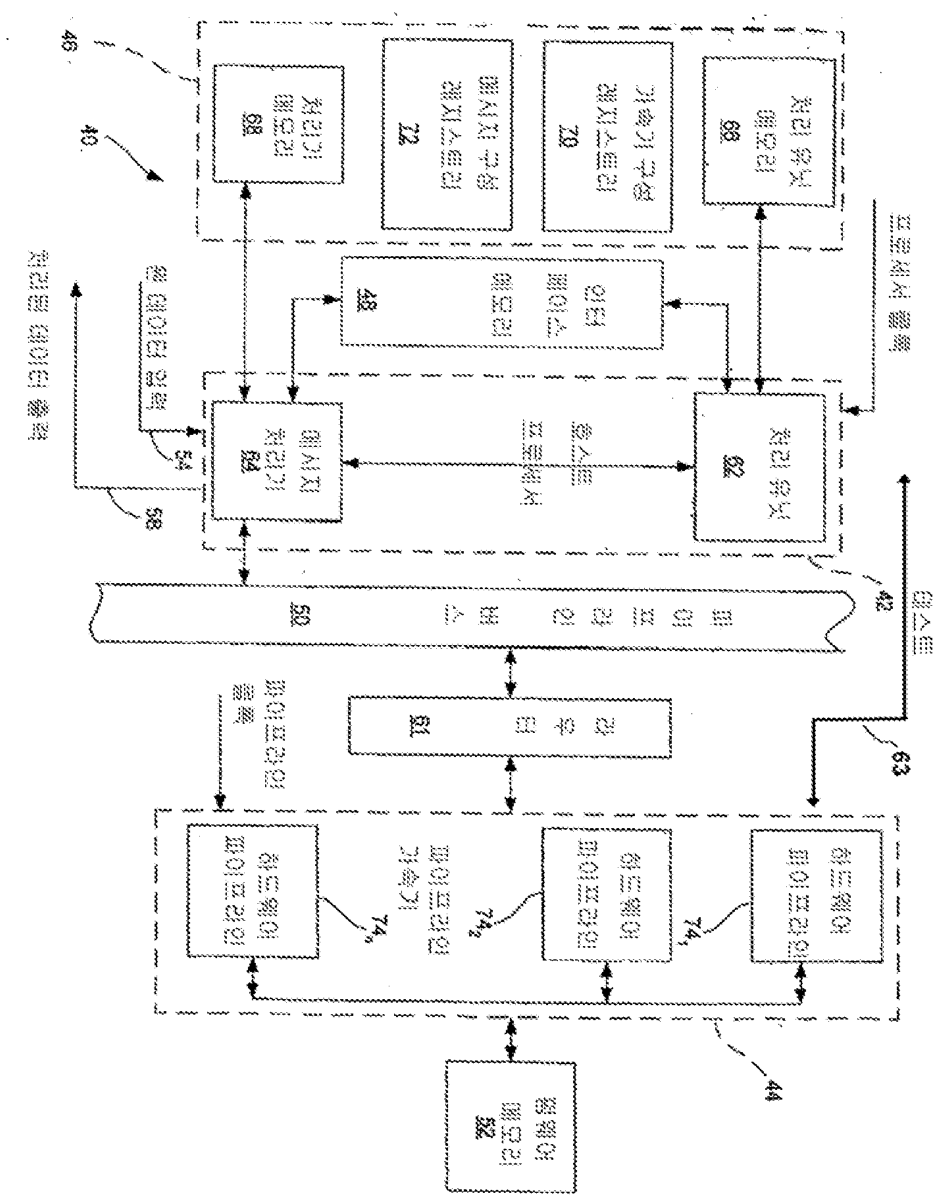
도면1


10
22
메모리
12
마스터-프로세서
26
분배 메모리
18
원 데이터 입력
20
처리된 데이터 출력
버스
16
14n
코-프로세서
14₂
코-프로세서
14₁
코-프로세서
24n
메모리
24₂
메모리
24₁
메모리

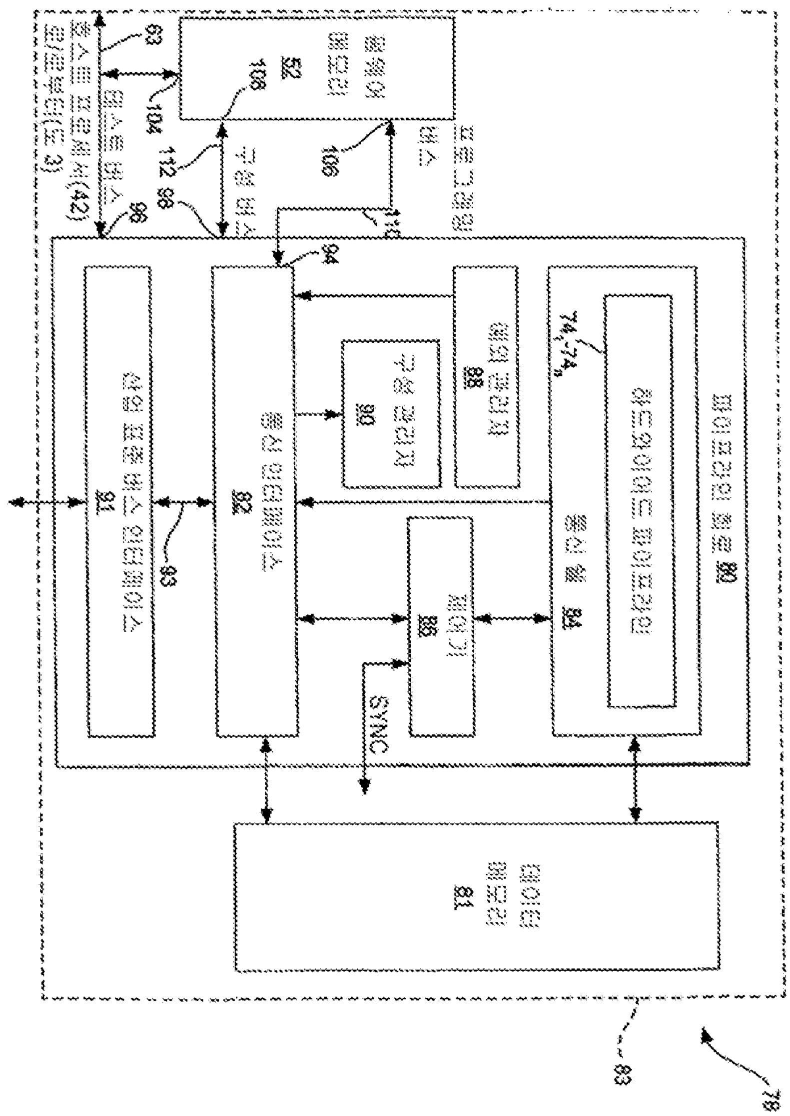
도면2



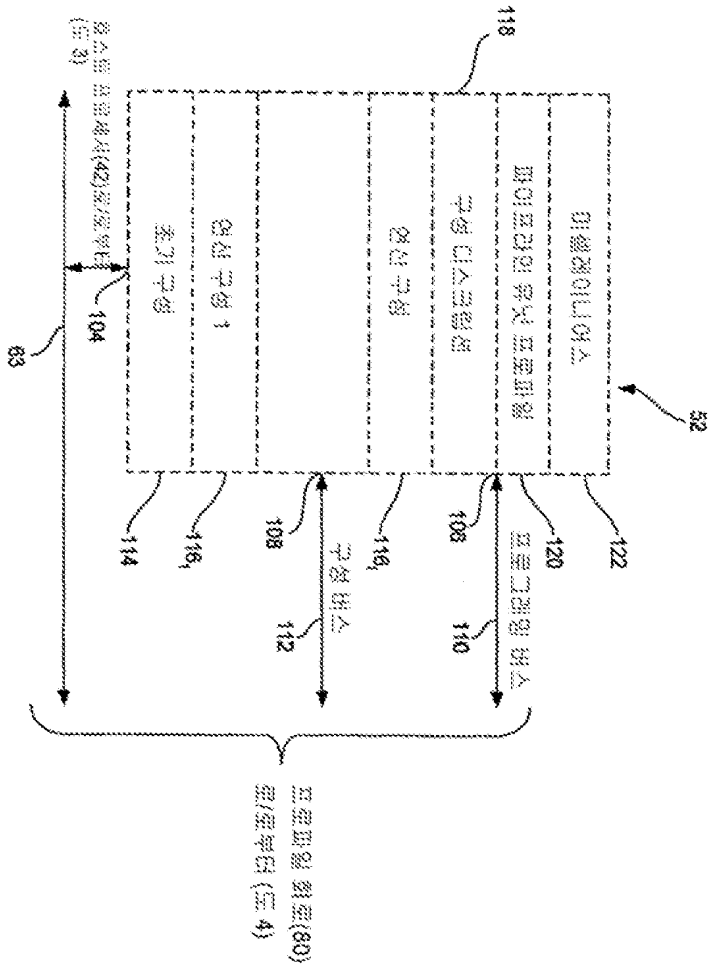
도면3



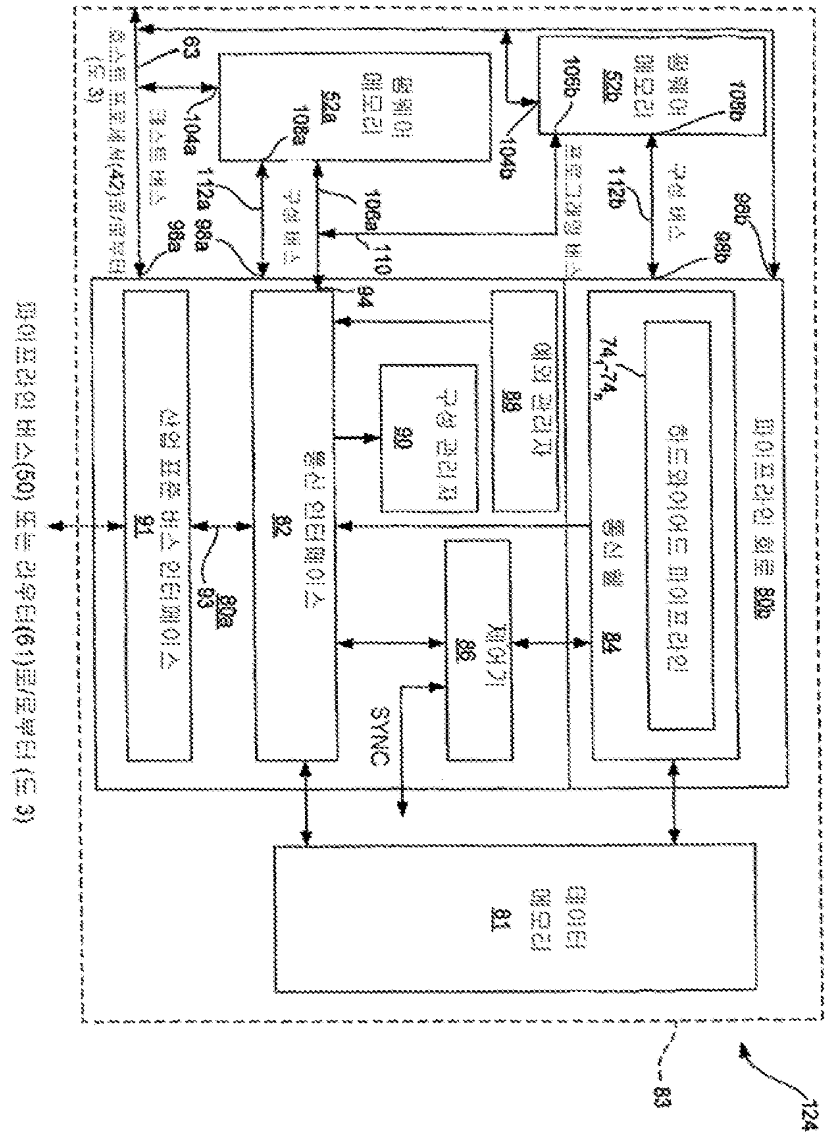
도면4


78
83
파이프라인 회로 80
하드와이어드 파이프라인
74₁-74ₙ
통신 쉘 84
예외 관리자
88
구성 관리자
90
제어기
86
SYNC
데이터
메모리
81
프로그래밍
버스
펌웨어
메모리
52
110
106
94
108
구성 버스
통신 인터페이스
82
93
63
104
112
98
96
테스트 버스
산업 표준 버스 인터페이스
91
호스트 프로세서(42)
로/로부터(도 3)

도면5


52
118
하드웨어 리소스
파워 프라임 유저 프로파일
구성 디스크립터
연산 구성
연산 구성 1
초기 구성
114
116₁
108
112
구성 버스
116ₙ
108
110
프로그래밍 버스
120
122
63
104
출력을 표시하기(42)위한 부분
(도 3)
프로파일 회로(80)
(도 4)

도면6


124
83
파이프라인 회로 80b
하드와이어드 파이프라인
74_1-74_n
통신 셸 84
데이터 메모리 81
예외 관리자 88
구성 관리자 90
제어기 86
SYNC
통신 인터페이스 82
80a
93
산업 표준 버스 인터페이스 91
펌웨어 메모리 52a
펌웨어 메모리 52b
108a
108b
106a
106b
104a
104b
110
112a
112b
98a
98b
94
63
구성 버스
프로그래밍 버스
호스트 버스
호스트 프로세서(42)로/로부터 (도 3)
파이프라인 버스(50) 또는 라우터(61)로/로부터 (도 3)